# EDM 2012

# 5th International Conference on Educational Data Mining

PROCEEDINGS OF THE
FIFTH INTERNATIONAL CONFERENCE ON
EDUCATIONAL DATA MINING

Chania, Greece, June 19-21 2012

**Kalina Yacef, Osmar Zaïane,**
**Arnon Hershkovitz, Michael Yudelson, John Stamper**

Kalina Yacef, Osmar Zaïane, Arnon Hershkovitz, Michael Yudelson and John Stamper

*International Conference on Educational Data Mining (EDM) 2012*
Proceedings of the 5th International Conference on Educational Data Mining
Kalina Yacef, Osmar Zaïane, Arnon Hershkovitz, Michael Yudelson and John Stamper (eds.)
Chania, June 19-21, 2012

ISBN: 978-1-74210-276-4

# Preface

The 5th International Conference on Educational Data Mining (EDM 2012) is held in picturesque Chania on the beautiful Crete island in Greece, under the auspices of the International Educational Data Mining Society (IEDMS). The EDM 2012 conference is a leading international forum for high quality research that mines large data sets of educational data to answer educational research questions. These data sets may come from learning management systems, interactive learning environments, intelligent tutoring systems, or any system used in a learning context.

We received a total of 50 full papers, 19 short papers and 20 posters from authors of 30 different countries and these submissions covered the full range of the conference topics specified in the call for papers. The papers were distributed to the 56 international program committee members, who called upon additional 17 external reviewers. All papers were reviewed by 3 reviewers and any paper submitted by organisers was sent to anonymous reviewers unbeknownst to the involved organisers. After receiving a total of 235 reviews, 17 full papers and 15 short papers were accepted to be presented at the conference and included in these proceedings, giving an acceptance rate of 34% for full papers, and 46% overall. Furthermore, 17 posters were accepted and their summaries appear in the proceedings. We have also included the abstracts of the 3 invited talks by (i) Professor Myra Spiliopoulou, Professor of Business Information Systems, Computer Science, Otto-von-Guericke University Magdeburg, Germany, (ii) Professor Danielle S. McNamara, Learning Sciences Institute Psychology Department, Arizona State University and (iii) Dr Bob Dolan, Senior Research Scientist, Assessment and Information, Pearson.

We would like to thank our sponsors Carnegie Learning (Gold Level), Pearson (Gold Level) and LearnLab (Silver Level) for their generous support. We would also like to thank the program committee members, the additional reviewers, the local committee, the web chair and the invited speakers for their invaluable help in putting this program and conference together.

We hope that the papers contained in these proceedings will be stimulating to most of the readers, provide thought-provoking new ideas to motivate new research, and prove helpful towards improving the literacy in educational data mining.

Enjoy the conference.

Kalina Yacef
Osmar Zaïane
Arnon Hershkovitz
Michael Yudelson
John Stamper

# Organization

## CONFERENCE CHAIR

| | |
|---|---|
| John Stamper | Carnegie Mellon University, USA |

## PROGRAM CHAIRS

| | |
|---|---|
| Kalina Yacef | The University of Sydney, Australia |
| Osmar Zaiane | University of Alberta, Canada |

## POSTER AND DEMO CHAIRS

| | |
|---|---|
| Arnon Hershkovitz | Worcester Polytechnic Institute, USA |
| Michael Yudelson | Carnegie Mellon University, USA |

## FINANCE AND WEB CHAIR

| | |
|---|---|
| Michael Bett | Carnegie Mellon University, USA |

## LOCAL ORGANIZING TEAM

| | |
|---|---|
| Kitty Panourgia | Neoanalysis, Greece |

## STEERING COMMITTEE

| | |
|---|---|
| Ryan Baker | Worcester Polytechnic Institute, USA |
| Tiffany Barnes | University of North Carolina at Charlotte, USA |
| Joseph E. Beck | Worcester Polytechnic Institute, USA |
| Michel Desmarais | Ecole Polytechnique de Montreal, Canada |
| Neil Heffernan | Worcester Polytechnic Institute, USA |
| Agathe Merceron | Beuth University of Applied Sciences, Germany |
| Mykola Pechenizkiy | Eindhoven University of Technology, Netherlands |
| Cristobal Romero | Cordoba University, Spain |
| Kalina Yacef | The University of Sydney, Australia |

## PROGRAM COMMITTEE

| | |
|---|---|
| Esma Aimeur | University of Montreal, Canada |
| Mirjam Augstein Kock | Upper Austria University of Applied Sciences, Austria |
| Elizabeth Ayers | University of California, USA |
| Tiffany Barnes | University of North Carolina at Charlotte, USA |
| Joseph Beck | Worcester Polytechnic Institute, USA |
| Bettina Berendt | Katholieke Universiteit Leuven, Belgium |
| Gautam Biswas | Vanderbilt University, USA |
| Jesus G. Boticario | U.N.E.D., Spain |
| Rafael A. Calvo | The University of Sydney, Australia |
| Min Chi | Stanford University, USA |
| Christophe Choquet | University of Maine, USA |
| Richard Cox | University of Edinburgh, Scotland |
| Sidney D'Mello | The University of Memphis, USA |
| Michel Desmarais | Ecole Polytechnique de Montreal, Canada |
| Mingyu Feng | SRI International, USA |

| | |
|---|---|
| Davide Fossati | Georgia Institute of Technology, USA |
| Eva Gibaja | Cordoba University, Spain |
| Daniele Godoy | Universidad Nacional del Centro de la Pcia, Buenos Aires, Argentina |
| Neil Heffernan | Worcester Polytechnic Institute, USA |
| Arnon Hershkovitz | Worcester Polytechnic Institute, USA |
| Roland Hubscher | Bentley University, USA |
| Sebastien Iksal | LIUM - University of Maine Computer Science Laboratory, Le Mans, France |
| Judy Kay | The University of Sydney, Australia |
| Jihie Kim | University of Southern California, USA |
| Evgeny Knutov | Technische Universiteit Eindhoven, The Netherlands |
| Kenneth Koedinger | Carnegie Mellon University, USA |
| Irena Koprinska | The University of Sydney, Australia |
| Vanda Luengo | Université Joseph Fourier, Grenoble, France |
| Tara Madhyastha | University of Washington, USA |
| Noboru Matsuda | Carnegie Mellon University, USA |
| Manolis Mavrikis | University of London, UK |
| Riccardo Mazza | University of Lugano/University of Applied Sciences of Southern Switzerland, Switzerland |
| Gordon Mccalla | University of Saskatchewan, Canada |
| Bruce Mclaren | Carnegie Mellon University, USA |
| Agathe Merceron | Beuth University of Applied Sciences Berlin, Germany |
| Julià Minguillón | Universitat Oberta de Catalunya, Spain |
| Jack Mostow | Carnegie Mellon University, USA |
| Kasia Muldner | Arizona State University, USA |
| Roger Nkambou | Université du Québec À Montréal, Canada |
| Alvaro Ortigosa | Universidad Autonoma de Madrid, Spain |
| Alexandros Paramythis | Johannes Kepler University, Austria |
| Zachary Pardos | Worcester Polytechnic Institute, USA |
| Mykola Pechenizkiy | Eindhoven University of Technology, The Netherlands |
| Cristobal Romero | Cordoba University, Spain Carolyn |
| Rose | Carnegie Mellon University, USA |
| Ryan S.J.D. Baker | Worcester Polytechnic Institute, USA |
| Erin Shaw | University of Southern California, USA |
| John Stamper | Carnegie Mellon University, USA |
| Jun-Ming Su | National Chiao Tung University, Taiwan |
| Steven Tanimoto | University of Washington, USA |
| Sebastian Ventura | Cordoba University, Spain |
| Stephan Weibelzahl | National College of Ireland, Ireland |
| Kalina Yacef | The University of Sydney, Australia |
| Michael Yudelson | Carnegie Mellon University, USA |
| Amelia Zafra Gómez | Cordoba University, Spain |
| Osmar R. Zaiane | University of Alberta, Canada |

## ADDITIONAL REVIEWERS

Iiya Goldin
I-Han Hsiao
Nan Li
Ming Liu
Collin Lynch
Roberto Martinez Maldonado
Terry Peckham
Hamman Samuel
Michael Sao Pedro
Mayya Sharipova
Carlos Soares
Mansoureh Takaffoli
Eric Verbeek
Qingyang Wang

## IEDMS OFFICERS AND BOARD OF DIRECTORS

| | |
|---|---|
| Ryan Baker (President) | Worcester Polytechnic Institute, USA |
| Michel Desmarais (Treasurer) | Ecole Polytechnique de Montreal, Canada |
| Tiffany Barnes | University of North Carolina at Charlotte, USA |
| Joseph E. Beck | Worcester Polytechnic Institute, USA |
| Neil Heffernan | Worcester Polytechnic Institute, USA |
| Agathe Merceron | Beuth University of Applied Sciences, Germany |
| Mykola Pechenizkiy | Eindhoven University of Technology, Netherlands |
| Cristobal Romero | Cordoba University, Spain |
| Kalina Yacef | The University of Sydney, Australia |

## Sponsors




http://www.carnegielearning.com
http://www.pearson.com
http://www.learnlab.org
http://www.educationaldatamining.org

# Table of contents

## Invited Talks (abstracts)

**Stream Mining in Education? Dealing with Evolution** **3**
*Professor Myra Spiliopoulou*

**From Text to Feedback: Leveraging Data Mining to Build Educational Technologies** **5**
*Danielle S. McNamara*

**Five Aspirations for Educational Data Mining** **7**
*Bob Dolan and John Behrens*

## Full Papers

**Assisting Instructional Assessment of Undergraduate Collaborative Wiki and SVN Activities** **10**
*Jihie Kim, Erin Shaw, Hao Xu and Adarsh G V*

**Automated Student Model Improvement** **17**
*Kenneth R. Koedinger, Elizabeth A. McLaughlin and John C. Stamper*

**Automatic Discovery of Speech Act Categories in Educational Games** **25**
*Vasile Rus, Arthur Graesser, Cristian Moldovan and Nobal Niraula*

**Co-Clustering by Bipartite Spectral Graph Partitioning for Out-of-Tutor Prediction** **33**
*Shubhendu Trivedi, Zachary Pardos, Gábor Sárközy and Neil Heffernan*

**Comparison of methods to trace multiple subskills: Is LR-DBN best?** **41**
*Yanbo Xu and Jack Mostow*

**Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models** **49**
*José González-Brenes and Jack Mostow*

**Identifying Learning Behaviors by Contextualizing Differential Sequence Mining with Action Features and Performance Evolution** **57**
*John Kinnebrew and Gautam Biswas*

**Identifying Students' Characteristic Learning Behaviors in an Intelligent Tutoring System Fostering Self-Regulated Learning** **65**
*François Bouchet, John Kinnebrew, Gautam Biswas and Roger Azevedo*

**Learner Differences in Hint Processing** **73**
*Ilya Goldin, Kenneth Koedinger and Vincent Aleven*

**Methods to find the number of latent skills** **81**
*Behzad Beheshti, Michel Desmarais and Rhouma Naceur*

**Mining Student Behavior Patterns in Reading Comprehension Tasks** **87**
*Terry Peckham and Gordon McCalla*

**Model-Based Collaborative Filtering Analysis of Student Response Data: Machine-Learning Item Response Theory** **95**
*Yoav Bergner, Stefan Droschler, Gerd Kortemeyer, Saif Rayyan, Daniel Seaton and David Pritchard*

**Predicting drop-out from social behaviour of students** **103**
*Tomas Obsivac, Lubos Popelinsky, Jaroslav Bayer, Jan Geryk and Hana Bydzovska*

**Searching for Variables and Models to Investigate Mediators of Learning from Multiple Representations** **110**
*Martina Rau and Richard Scheines*

**The Impact on Individualizing Student Models on Necessary Practice Opportunities** **118**
*Jung In Lee and Emma Brunskill*

**Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra** **126**
*Ryan S.J.D. Baker, Sujith Gowda, Michael Wixon, Jessica Kalka, Angela Wagner, Aatish Salvi, Vincent Aleven, Gail Kusbit, Jaclyn Ocumpaugh and Lisa Rossi*

**Using Edit Distance to Analyse Errors in a Natural Language to Logic Translation Corpus** **134**
*Dave Barker-Plummer, Robert Dale, Richard Cox and Alex Romanczuk*

## Short Papers

**Calculating Probabilistic Distance to Solution in a Complex Problem Solving Domain** **144**
*Leigh Ann Sudol, Kelly Rivers and Thomas K. Harris*

**Classification via clustering for predicting final marks starting from the student participation in Forums** **148**
*Manuel Ignacio Lopez, Cristobal Romero, Sebastián Ventura and J.M. Luna*

**Development of a Workbench to Address the Educational Data Mining Bottleneck** **152**
*Ma. Mercedes Rodrigo, Ryan S. J. D. Baker, Bruce McLaren, Alejandra Jayme and Thomas Dy*

**Early Prediction of Student Self-Regulation Strategies by Combining Multiple Models** 156
*Jennifer Sabourin, Bradford Mott and James Lester*

**Identifying Successful Learners from Interaction Behaviour** 160
*Judi Mccuaig and Julia Baldwin*

**Interaction Networks: Generating High Level Hints Based on Network Community Clusterings** 164
*Michael Eagle, Matthew Johnson and Tiffany Barnes*

**Interleaved Practice with Multiple Representations: Analyses with Knowledge Tracing Based Techniques** 168
*Martina Rau and Zachary Pardos*

**Learning Gains for Core Concepts in a Serious Game on Scientific Reasoning** 172
*Carol Forsyth, Philip Pavlik Jr, Arthur Graesser, Zhiqiang Cai, Mae-Lynn Germany, Keith Millis, Heather Butler, Diane Halpern and Robert Dolan*

**Leveraging First Response Time into the Knowledge Tracing Model** 176
*Yutao Wang and Neil Heffernan*

**Meta-learning Approach for Automatic Parameter Tuning: A case of study with educational datasets** 180
*María De Mar Molina, Cristobal Romero,Sebastián Ventura and J.M. Luna*

**Mining Concept Maps to Understand University Students' Learning** 184
*Jin Soung Yoo and Moon-Heum Cho*

**Policy Building – An Extension To User Modeling** 188
*Michael Yudelson and Emma Brunskill*

**The real world significance of performance prediction** 192
*Zachary Pardos, Qing Yang Wang and Shubhendu Trivedi*

**The Rise of the Super Experiment** 196
*John Stamper, Derek Lomas, Dixie Ching, Steven Ritter, Kenneth Koedinger and Jonathan Steinhart*

**Using Student Modeling to Estimate Student Knowledge Retention** 200
*Yutao Wang and Joseph Beck*

## Posters

**A promising classification method for predicting distance students' performance.** 206
*Diego García-Saiz and Marta Zorrilla*

**Analyzing paths in a student database** **208**
*Donatella Merlini, Renza Campagni and Renzo Sprugnoli*

**Analyzing the behavior of a teacher network in a Web 2.0 environment** **210**
*Eliana Scheihing, Carolina Aros and Daniel Guerra*

**Automated Detection of Mentors and Players in an Educational Game** **212**
*Fazel Keshtkar, Brent Morgan and Arthur Graesser*

**Categorizing Students' Response Patterns using the Concept of Fractal Dimension** **214**
*Rasil Warnakulasooriya and William Galen*

**CurriM: Curriculum Mining** **216**
*Mykola Pechenizkiy, Nikola Trcka, Paul De Bra and Pedro Toledo*

**Data mining techniques for design of ITS student models** **218**
*Ritu Chaturvedi and Christie Ezeife*

**Deciding on Feedback Polarity and Timing** **220**
*Stuart Johnson and Osmar Zaiane*

**Finding Dependent Test Items: An Information Theory Based Approach** **222**
*Xiaoxun Sun*

**Fit-to-Model Statistics for Evaluating Quality of Bayesian Student Ability Estimation** **224**
*Ling Tan*

**Inferring learners' knowledge from observed actions** **226**
*Anna Rafferty, Michelle Lamar and Thomas Griffiths*

**Learning Paths in a Non-Personalizing e-Learning Environment** **228**
*Agathe Merceron, Liane Beuster, Margarita Elkina, Albrecht Fortenbacher, Leonard Kappe, Andreas Pursian, Sebastian Schwarzrock and Boris Wenzlaff*

**Similarity Functions for Collaborative Master Recommendations** **230**
*Alexandru Surpatean, Evgueni Smirnov and Nicolai Manie*

**Social Networks Analysis for Quantifying Students' Performance in Teamwork** **232**
*Pedro Crespo and Claudia Antunes*

**Speaking (and touching) to learn: a method for mining the digital footprints of face-to-face collaboration** **234**
*Roberto Martinez Maldonado, Kalina Yacef and Judy Kay*

**Stress Analytics in Education** **236**
*Rafal Kocielnik, Mykola Pechenizkiy and Natalia Sidorova*

**Variable Construction and Causal Discovery for Cognitive Tutor Log Data: Initial Results** **238**
*Stephen Fanesali*

**List of Authors** **241**

# INVITED TALKS

(abstracts)

# Stream Mining in Education? Dealing with Evolution

Prof. Myra Spiliopoulou, Knowledge Management and Discovery Lab KMD, Faculty of Computer Science, Otto-von-Guericke-University Magdeburg, Germany

EDM methods that suggest materials to students are based on student models, and/or on the behavior of other, similar students. Whenever a recommendation is formulated, though, or the similarity between two students is assessed, each of the observed individuals is at some, a priori unknown and typically not observable intermediate state of the learning process. This process manifests itself as a drifting stream of activities. Learning methods that adapt their models to the current state of this stream allow the formulation of recommendations aligned to the current learning stage of a student, taking into account students that evolve/learn similarly to her. The underlying technology is that of stream mining, rather than data mining.

This talk is on the potential and challenges of conventional stream mining and relational stream mining for educational purposes. We start with mining over a conventional stream of activities, such as the interaction with a platform containing learning materials. The typical learning task is predicting the next course material, given the changes in the preferences and experience of the individuals. Then, we move over to the learning task of discovering groups of individuals that evolve similarly over time. Finally, we discuss the supervised task of learning a model of performance, taking into account that the performance of each individual may change (unexpectedly) during the observation process.

The examples of this talk do not come from the field of Educational Data Mining but from closely related fields -- formulating recommendations for products, and model adaptation as people's behavior change.

# From Text to Feedback: Leveraging Data Mining to Build Educational Technologies

Danielle S. McNamara, Arizona State University

Over the past decade, our research teams have been using data mining techniques primarily for the purpose of analyzing text and writing corpora. These analyses emerged within three projects, Coh-Metrix, iSTART, and the Writing-Pal. Coh-Metrix is a text analysis tool that provides hundreds of linguistic and semantic indices on text. Within the context of the Coh-Metrix project, we have analyzed thousands of texts with the goal of better understanding the nature of text. Ultimately, our goal has been to provide educators with multidimensional information about the difficulty of text. iSTART is a reading strategy tutoring system that provides students with automated feedback on the quality of their self-explanations of text. The Writing Pal is a writing strategy tutoring system that provides strategy instruction to adolescent readers as well as practice writing essays with feedback. Within the context of the iSTART and Writing Pal projects, we have used Coh-Metrix and other text analysis tools to analyze self-explanations and essays. Our goal has been to better understand the linguistic features contributing to their quality as well as to develop and improve our automated scoring and feedback systems. All of these endeavors have called upon a variety of data mining techniques that serve to analyze the data and in turn drive the feedback algorithms that undergird the tutoring systems. This talk will describe how data mining supports the development of educational technologies at various levels.

# Five Aspirations for Educational Data Mining

Bob Dolan and John Behrens, Pearson

In this talk we present our disciplinary, methodological, and social aspirations for Educational Data Mining. These aspirations are based on broad conceptualizations of the nature of education and data analysis as social endeavors. These aspirations fall into five categories which begin with the statement that "We hope that Educational Data Mining will…" 1) Consider the broad range of the social and organizational aspects of education and its administration, including informal and ubiquitous learning; 2) Consider the broad range of inputs of digital artifacts that feed into the design of learning systems (not just the outcomes of system interactions); 3) Consider data mining as a human endeavor which is itself a proper topic of psychological, sociological and other academic disciplines; 4) Remember the fundamentals of quality data analysis regardless of computational techniques (with special fondness for John Tukey's insights); 5) Provide information that celebrates the diversity of effective pedagogies and supports learning by the outliers, hidden clusters, and otherwise missed special groups of people that are lost in the averages or other insensitive aggregations. We have high expectations for the field of Educational Data Mining to evolve broadly and contribute broadly to education and society.

# LONG PAPERS

# Assisting Instructional Assessment of Undergraduate Collaborative Wiki and SVN Activities

Jihie Kim, Erin Shaw, Hao Xu, Adarsh G V
Information Sciences Institute / University of Southern California
4676 Admiralty Way, Marina del Rey CA 90292
{jihie, shaw, hxu, agvenkat}@isi.edu

## ABSTRACT

In this paper we examine the collaborative performance of undergraduate engineering students who used shared project documents (Wikis, Google documents) and a software version control system (SVN) to support project collaboration. We present an initial implementation of TeamAnalytics, an instructional tool that facilitates the analyses of the student collaboration process by creating dynamic summaries of team member contributions over time in. Document content is processed using machine learning techniques. We validated the summary's effectiveness using a questionnaire given to instructors and team managers. Team managers indicated that summaries of student contribution to coding activities influenced their evaluation and coordination of team projects.

## Keywords

Collaborative teamwork, instructor tool, wiki, team projects.

## 1. INTRODUCTION

Engineering students participating in collaborative activities communicate electronically through a variety of applications, most of which are inaccessible to an instructor and thus offer little insight into the process of collaboration. The goal of the Pedagogical Wiki project is to assist instructors and educational researchers in evaluating team and individual student performance in the context of computer-supported collaborative learning environments.

In this paper we examine the collaborative performance of undergraduate engineering students who used a shared project documents, including Wikis and Google documents, and a software version control system to support project collaboration. Wikis are editable Web sites that support the creation of linked pages, archiving of media, revision control, access control, searching, and a consistent look and feel. Wikis facilitate collaborative learning by allowing groups of laypersons to collaboratively create web content [13,1,4]. However, the research on the effectiveness of using Wikis for student collaboration has been mixed [14,17], and patterns of student collaborative documenting and their effect on learning have not been fully assessed.

In addition to Wikis, students used Google documents, a popular team document generation and sharing environment that allows synchronous document editing, and Subversion (SVN), a version control system that is commonly used for software management. Version control systems track revisions that are made to files over time, usually by a group of authors. Wikis, Google documents and SVN all provide revision "histories", which can, in theory, be used to analyze student performance. For example, Ben-Zvi [1] notes that while logs can be used to evaluate each student's Wiki contribution, the number of contributions is enormous and new techniques and tools are needed to track them efficiently. Without proper tools, the analysis of document histories would place a considerable burden on instructors, who rarely have the skills or time to analyze the data for assessment purposes.

This paper presents a new instructional tool called *TeamAnalytics* that summarizes collaboration via online team activities. It dynamically processes student shared document edits and code management actions, summarizes both the overall team and individual contributions in each week, and presents the summary to the team managers and the instructor. For processing Wiki content data, we use natural language processing (NLP) techniques and machine learning approaches to generate topic-based summary of the documents. We report a study of PedSummary based on team manager ratings and a small survey. The initial results with two undergraduate courses with large team projects indicate that individual code contribution summary is useful for team managers and such summary can influence how the managers coordinate the team project.

### 1.1 Teamwork summary categories

**Table 1.** Current categories of team work summaries.

| Category | Summary | Description |
|---|---|---|
| 1 View Docs | A tree view of Wiki docs | Hierarchical view of the docs, organized based on links and topics |
| 2 Wiki-Group | Total Wiki contributions by members | Number of docs (and number of words) created, frequencies of accesses/views, and topic-based document distribution |
| 3 Wiki-Individual | Individual Wiki contributions | Number of docs, words, access/view of others' docs and topic-based distribution of the docs |
| 4 SVN-group | Total SVN contributions by members | Number of files added/modified/deleted by the team, and weekly totals |
| 5 SVN-Individual | Individual SVN contribution | Number of files added/modified/deleted by the student, and weekly totals |

As engineering researchers, we (the authors) use Wikis extensively, primarily as a knowledge repository for project documentation and media. It is clear that, for Wikis, the benefit of democratic use is also its downfall, with its lack of structure and

oversight. Student Wiki sites often do not scale well, and tracking text and asset contributions becomes frustrating. Our goal was to provide finer-grained measurements and user-friendly interfaces for understanding instructional shared Wiki use.

In order to alleviate the problem of viewing student documents with existing Wiki systems, TeamAnalytics clusters documents according to the link hierarchy within the Wiki system (Category 1 in Table 1). They are also organized based on document topics that are automatically identified from topic models.

The primary instructor for the undergraduate course we studied also teaches an upper level course in which the students take on the role of team managers for the undergraduate project teams, and the instructor delegates most of the assessment tasks for the project teams to the team managers. A needs assessment was performed for both the course instructor and the team managers. Although the team managers participate in student group meetings and help the students as needed, often times they had difficulty in documenting who is doing what and how much. Such manager documentation is used in reporting teamwork to the instructor and tracking the teamwork throughout the project. In some cases, the members in the same team receive similar grades depending on the team performance. The instructor and the managers wanted to see individual contributions as well as the total contributions by all the team members (Categories 3 and 5 vs. 2 and 4 in Table 1). Identifying patterns of student activity relative to student performance was also discussed. In order to support an analysis of activity patterns, we broke up the contributions into weekly activities so that the managers can see how students work towards the deadlines over time.

## 2. ANALYSIS REPORT GENERATION

This section describes how collaborative Wiki and Google document activities are captured into a summary that is viewable by team managers and instructors. Although we show results for Moodle's Wiki, Google documents and SVN, most of the data processing steps do not depend on the course management system or particular document tools. For example, our topic classification functions are being used for other Wiki (e.g. Brainkeeper) content.

### 2.1 Participating Courses

The TeamAnalytics system was integrated into Moodle's [10] virtual learning environment during the Spring 2011 and Fall 2011 semesters. During each of these semesters, two undergraduate software engineering courses were combined for a large team-based project assignment. The study took place at the University of Southern California.

Students in a freshman level software development course (CSCI200) teamed up with students in a sophomore level course (CSCI201) for a large-scale programming project. Students in both courses learned team management, software engineering principles, and operating system principles and used the concepts to build "authentic" applications that solved new problems. Because second year students had already completed the first year course, they were able to mentor the first year students. The project team had students from both classes. Each team had about four freshmen and four sophomore students. The first year course (CSCI200) emphasized user-interfaces and the second year course focused on architecture (CSCI201). Additionally, a team manager was assigned to each team to assesses team co-ordination and leadership skills, and provide help throughout the project. Our work focused on assisting the team managers and the instructors.

There were ten teams of between ten and fifteen students each semester. The teams used their collaborative workspaces (Moodle) in myriad ways. Some teams used the Moodle Wiki and some used Google documents that they then linked to the Moodle courses. Some used a combination of both, e.g. Wikis for meeting notes and Google for documents. The choice was theirs. The workspace for team M2, is shown in Figure 1.

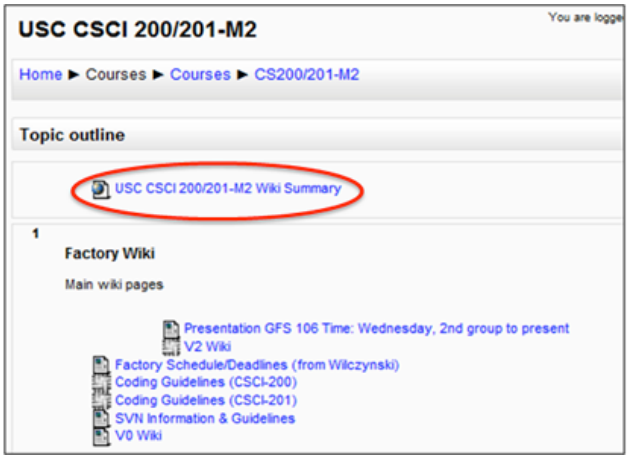



**Figure 1.** The collaborative workspace for a combined USC freshman/sophomore engineering team M2

### 2.2 Data processing

The TeamAnalytics architecture is shown in Figure 2. All team activity data is stored in the Student Group Activity database. The system fetches SVN activity data from the SVN server used by the courses. Students' actions including addition, modification and deletion of files are retrieved every 24 hours.

The system also dynamically accesses the student Wiki history including addition, deletion and modification information from the course management system. Each team provided edit permissions to allow us to access to the content and edit history of the shared Google documents through a Google API. After re-formatting the data, the Wiki data processing functions were used.

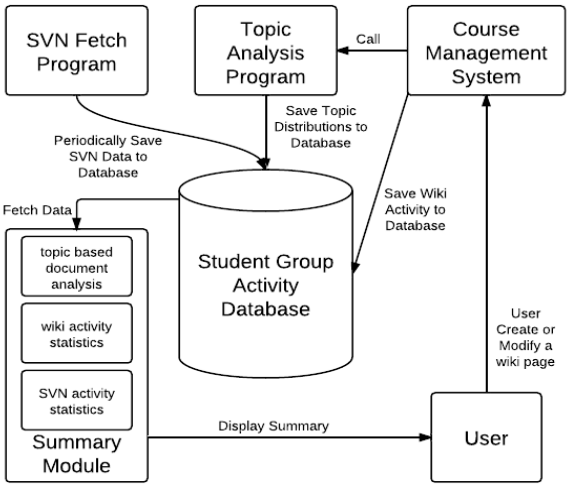



**Figure 2.** Generating teamwork summary using data from SVN and Wikis and Google documents.

For topic modeling, whenever a new page or a revised page is saved, a backend program is invoked to parse the content of the page and generate topic distributions using the automatic topic classifier, which is described in detail in subsequent sections.

The dynamically generated summaries were sent to the team managers and instructors by weekly email. The summaries were also viewable from within the team's Moodle course environment. The team manager of M2 could access the summary by clicking the 'USC CSCI200/201-M2 Wiki Summary' link (Figure 1). The instructor and the team managers could view all the teams' activities as shown in Figure 3. The content of the summaries is described in Section 3.

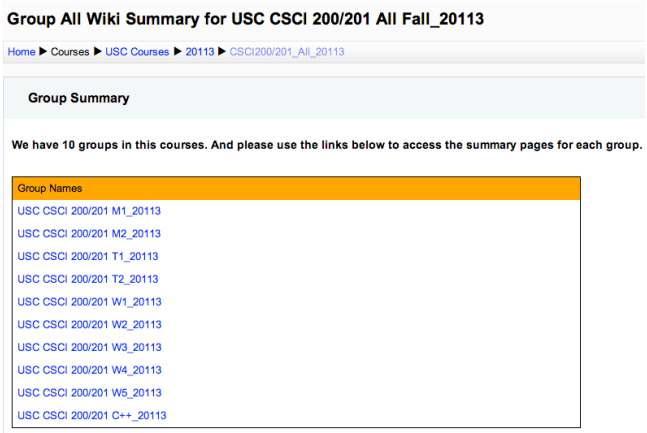



**Figure 3** Summaries of all the participating teams available for the instructors and team managers.

## 2.3 Automatic topic classification

The Wiki pages and Google docs are classified based on the page title and the content using Labeled LDA.

### *2.3.1 Background on Labeled LDA*

Because we wanted to develop a topic modeling approach that could be easily applied to different courses, supervised approaches requiring a large amount of labeled data were not appropriate. And because discussion datasets are noisy, we needed a model that could capture semantic meanings behind the words rather than words themselves. LDA (Latent Dirichlet allocation) 32] is very powerful in analyzing latent topics of documents, but it has all the disadvantages inherent to any unsupervised model. The topic distribution of LDA depends on the word distribution in the documents and cannot be controlled even if we have a prior knowledge to guide topic generation. Thus many topics are just a cluster of words that co-occur in many documents and do not have a semantic meaning in real data. Ramage et al [13] introduced a semi-supervised algorithm, called Labeled LDA, a novel model that uses multi-labeled corpora to address the credit assignment problem. Unlike traditional LDA, Labeled LDA constrains topics of documents to a given label set. We have V number of unique vocabularies and D number of documents, and K number of topics. For each document d, which consists of a list of word $(w_1(d),\ldots,w_N(d))$, we have k dimensional binary topic indicators. Unlike using symmetric Dirichlet distribution with a single hyper parameter $\alpha$ as a Dirichlet prior on the topic distribution $\theta(d)$, Labeled LDA restricts $\theta(d)$ to only over the topics that correspond to observed labels. The key task was to select a label set that could generate meaningful topic results.

### *2.3.2 Wiki Topic Modeling with Labeled LDA*

The topic categories for the software engineering team wiki documents are shown in Table 2. This was generated after manual analyses of the course curriculum and the content of the wiki documents across all the project groups in the class. The topic categories represent the major types of the documents generated by the students over the course. The two main topic classes are team management categories (Team Organization and Progress Summary). The rest of them represent software engineering principles documents that show Initial Planning, Design, Coding, Testing and System Analysis.

**Table 2.** Topic categories for team work document.

| Topics | Description | Kappa |
|---|---|---|
| **Team Org**anization | Team contact information, availability for meetings, allocation of project modules to members, milestones and their target dates. | 0.83 |
| Initial **Planning** | Initial research and Q&A discussions on the project. Some use external links and reference materials. | 0.82 |
| **Design** | Design documents describing frontend, backend, Interfaces. Also includes interactive diagrams. | 0.73 |
| Implementatio n and **Coding** | Describes implementation method, using pseudo code or java code. | 0.91 |
| **Testing** | Describe code testing including unit test cases, bugs or Junit. | 0.73 |
| System **Analysis** | Analysis of implemented system; system issues, setbacks and fixes in system design. | 0.74 |
| **Progress** Summary | Document discussions, meetings, email and phone conversations on the status of project modules. | 0.81 |

A Kappa measure [5] was used to verify agreement. Table 2 shows the Kappa values between two annotators for 263 documents sampled. Kappa values take into account agreement that can occur by chance.

$$\text{Kappa} = \frac{\text{Observed agreement - Chance agreement}}{\text{Total observed - Chance agreement}}$$

**Table 3.** Sample label set and LLDA classification results.

| Topics | Sample Labels | % | Prec/re call |
|---|---|---|---|
| Team Org | rxgui,rxnub,part,kit,on,panel,gantri,rxnub_rxnub,r xfunction,object,return,user,factori,move,can | 14.8 | 0.4 /0.29 |
| Planning | rxquestionmark,custom,=,{,},messag,if,rxfunctio n,order,state,list,check,action,customer,timer | 3.5 | 0.0 /0.0 |
| Design | kit,part,public,rxnub,rxfunction,int,void,on,stand, rxquestionmark,type,public_void,call,=,at | 28.3 | 0.43 /0.35 |
| Coding | rxfunction,=,part,{,rxnub,},if,void,int,public,kit,s et,bin,rxgui,feeder | 19.0 | 0.69 /0.82 |
| Testing | },{,part,=,kit,rxfunction,if,nest,int,return,rxquesti onmark,lane,void,true,list | 9.4 | 0.0 /0.0 |
| Analysis | rxgui,you,panel,rxnub,food,cook,on,ar,when,anim ,should,do,if,all,your | 6.3 | 0.54 /1.0 |
| Progress | rxnub,rxnub_rxnub,i,test,meet,task,rxday,my,inte gr,code,on,rxgui,done,design,panel | 16.7 | 0.65 /1.0 |

Sample label sets used for LLDA are show in Table 3. We evaluated the model distributions using the manual annotations as the gold standard. Since documents can contain multiple topics, we evaluated them by selecting and comparing the top 2 topics

from the manual annotations and model results. Precision is defined as the ratio of the number of correct topic annotations generated by the model to the total number of topic annotations generated by the model. Recall is defined as the ratio of the number of correct topic annotations generated by the model to the total number of correct annotations specified by the gold standard. The table also shows the % of the topics within 314 annotated documents. The current model provides limited accuracies for some topic categories due to limited examples. We are currently improving the LLDA model by adding more dataset.

# 3. ANALYSIS REPORT PRESENTATION

This section describes how document-based and code-based activity summaries were presented to team managers and instructors. As described above, the dynamically updated summaries and statistics were viewable from within Moodle. We also generated and sent team summary reports to each team manager by email.

## 3.1 Document Summary

This section describes the content of the document summaries.

### *3.1.1 Tree view of document with topic labels*

A tree view of the documents created or modified by students on team W3 is shown in Figure 4. Each team generated more than hundred documents and uploaded many additional files such as design diagrams. Wiki pages, plan text pages, and upload documents of any type were stored within the virtual learning environment. Wiki pages were related using hyperlinks. Google documents were also used and linked within Moodle. In order to help students and team managers navigate through various documents, TeamAnalytics compiled document links and generated a hierarchical view of the team documents. A general API (application programming interface) was developed so that other types of links could be captured within the structure.

The tree view also shows who created the document, how many students edited the document, how many edits were made, how long the document was edited, how many words were included, and how many links were present in the document. We also organized the documents based on the content topics, using the above-mentioned LLDA models. Without reading the individual document details, team managers could evaluate who was contributing on what topic and how often.

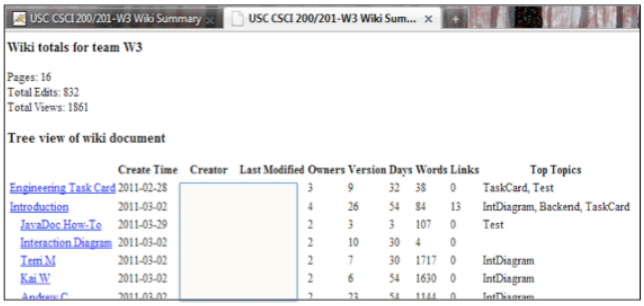
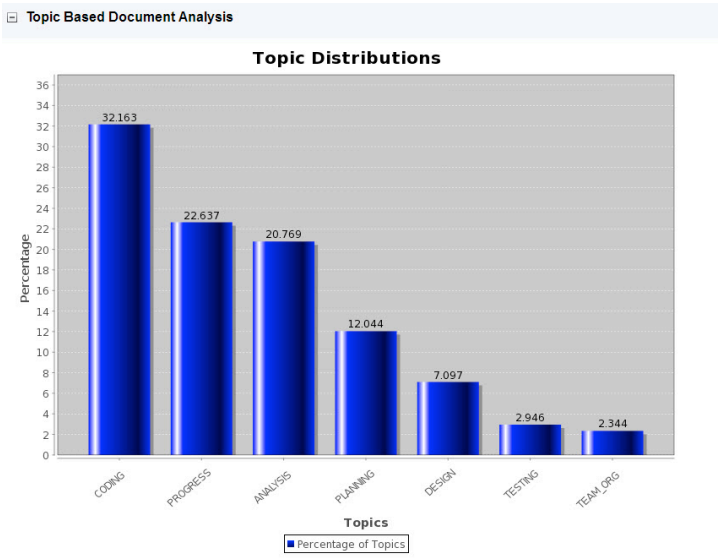
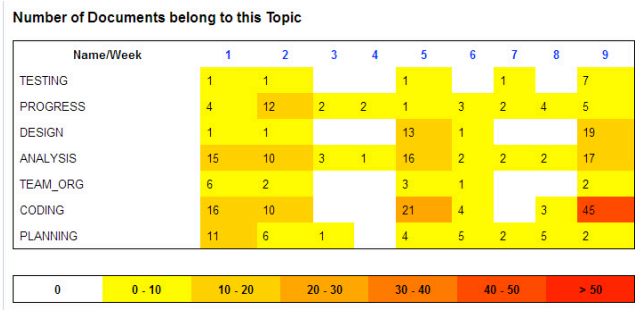



**Figure 4.**Tree view of documents based on document links

### *3.1.2 Topic based document distribution*




**(a)** Topic distribution of all the team documents.




**(b)** Weekly document topic distribution of documents.

**Figure 5.**Topic distribution of team documents.

Document topics were summarized into a bar graph like the one shown in Figure 5-(a). The accumulated number of documents per each topic, based on the LLDA topic distribution, is shown. Using this view, the team managers could estimate the distribution of topics in the team documents. We also highlighted increments within a given week so the viewer could evaluate the topics of focus during that week. A weekly distribution of the document topics is shown in the heat map in Figure 5-(b). The headings 1-9 depict the nine weeks that the project runs. The cells with high frequency values are highlighted with darker colors.

### *3.1.3 Participation frequency per student*

Wiki contributions by individual students are shown in Figure 6. For each student the left (blue) bars show number of documents viewed and the right (green) bars show the number of documents edited by the student. The portions contributed during the current week are highlighted with lighter colors, and the counts at the tops of the bars show the current week's numbers of edits and views.

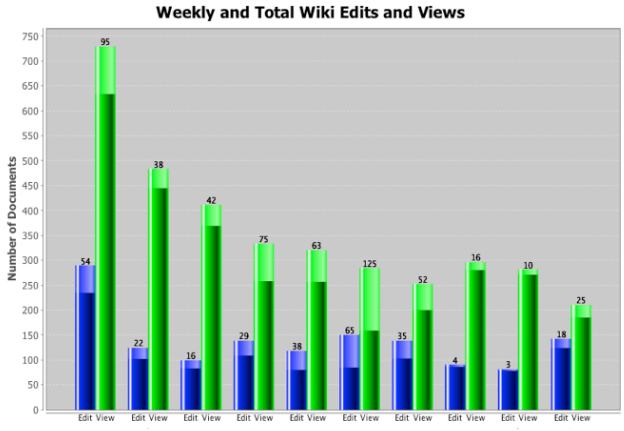
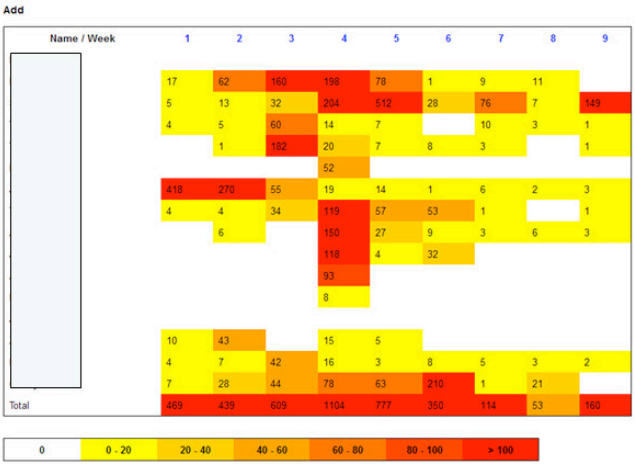
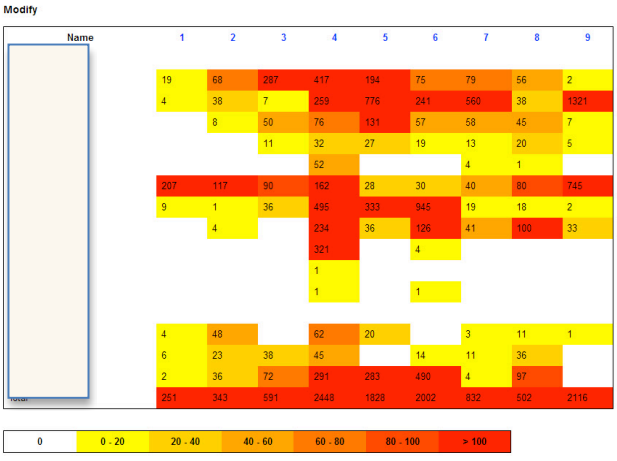



**Figure 6.** Individual student contributions to Wiki

## 3.2 SVN Summary




**(a)** Number of files added by each student weekly.




**(b)** Number of files modified by each student weekly.

**Figure 7.** Weekly student contributions to SVN

Students used the Subversion (SVN) version control system to manage changes to their team's programming files. SVN allowed team members to add new program files, or modify or delete existing ones. Figure 7 shows individual student contributions to SVN for adding and modifying files. The weekly total numbers of file additions and modifications by all the team members are shown at the bottom of the table. The team managers were able to track the degree of SVN activity using this summary.

# 4. USER STUDY

TeamAnalytics was integrated into Moodle's virtual learning environment during the 2011 Spring and Fall semesters. A total of 278 students participated in the projects (42 freshmen and 67 sophomores in the Spring implementation, and 90 freshmen and 57 sophomores in the Fall implementation). There were ten teams each semester, and a manager was assigned for each team. The system was introduced to the classes and team managers before the project started. The dynamic summary was available to team managers on Moodle, and also sent weekly by email.

**Table 3.** Team manager ratings of the TeamAnalytics components.

| Team Manager Rating (average) | Spr 2011 (N=7) | Fall 2011 (N=7) |
|---|---|---|
| Viewing: 1=never, 2=occasionally, 3=often | | |
| Document activity viewed | 2.0 | 1.71 |
| Topic activity viewed | -- | 1.57 |
| SVN activity viewed | 2.7 | 2.29 |
| Helpfulness: 1=minimal, 2=moderate, 3=very | | |
| Document activity helpful/ influence manager | 1.9 | 2 |
| Topic activity helpful/ influence manager | -- | 1.6 |
| SVN activity helpful/ influence manager | 2.4 | 2.8 |

Survey responses for the team managers are shown in Table 4. Survey participation was voluntary and the response rate for both semesters was seven out of ten. Team managers were asked to rate the document (Wiki and Google Docs) activity views, topic-based document summaries and SVN activity summaries separately. The topic-based document summary was developed later and introduced to the Fall 2011 classes only.

The team managers viewed the SVN activity summaries more often than the document and topic summaries, and found the SVN summaries between moderately and very helpful. The documents summaries were rated moderately helpful and the topic summary was rated between minimal and moderate. It is evident that the team managers were most interested in student coding activities.

Team manager responses to other survey questions are shown in Table 5. The managers liked how they could keep track of coding work progress using TeamAnalytics. Several managers raised issues about the user interface (UI) especially comparing the old and new Moodle UIs. Recent upgrades and our own improvement of the interface design reduced some of the concerns. Individual managers show different preferences for how the information should be presented. We are investigating alternative approaches for showing the results. The managers also wanted to see more

details on student coding activities such as the numbers of lines added or deleted by individual students. We plan include such coding activity information and provide a drill-down view where end users can choose to see such details

**Table 5.** Team manager answers for survey questions.

| Question | Responses | |
|---|---|---|
| How did the summary influence your work? | s p | showed me the progress of the team |
| | | It did influence the way I saw the member and the team as I knew who was putting in more effort. This is definitely better than the views and edits |
| | | I will review and evaluate team members' performances for their projects. |
| | | Frequency of SVN commits does not map to work done on the project, but low commits CAN be a reason for concern |
| | | Allowed for me to monitor which members were contributing to code. |
| Can you think of additional activity information that you want to see? | s p | Perhaps the time of the last edit or view |
| | | Member wiki contribution statistics |
| | | I like it how it is |
| | f a | If it's possible, analysis showing number of lines of code affected, number of files touched, whether the commit is a merge. Any of those would be helpful for analysis |
| | | I collected activity data by being at the meetings, so no. |
| | | Meeting set up and attendance checking. |
| | | Individual doc revision statistics for each document |
| | | The question of are we behind, maybe by comparing with SVN commits of previous semesters? |
| | | Whether someones work has actually been productive |
| Do you have any suggestions for improving the summary, including how the information is displayed? | s p | The current way is good enough as it serves the purpose, The UI should be improved |
| | | In graph instead of tables. Better interface |
| | | moodle's UI sucks |
| | | you guys were great with feedback this semester and the moodle was FAR more useful than it was in the fall of 2010. I like it how it is. |
| | | In a dashboard format where priority is given to certain issues. Bug tracking. |
| | | Information would be more easily shared if it was in more of a forum type setup. |
| | f a | Allow more detail in the SVN log, like separate it by day or hour (on a graph) instead of week. Also stronger analysis like graph of SVN activities for each person. |
| | | Less colored charts, more bar graphs |

## 5. RELATED WORK

Our work is situated in the research domain of context modeling and activity awareness to support group performance on complex tasks (e.g., [3,18]). Of particular relevance is Upton and Kays' Narcissus system [16], which graphically models user and group behavior to support team collaboration. Also related is Suthers, Dwyer and Medina's [15] Uptake Analysis Framework for conceptualizing and representing distributed interaction, in which contingency graphs are used to transcribe activity in the temporal space, distributed across multiple documents, to enable researchers to (possibly) identify the influence of prior activity on ongoing activity. Our work extends existing research by automatically generating summaries of group work in collaborative knowledge building and team programming environments, and by combining NLP techniques to support topic-based analysis of contribution content.

Our work builds on Activity Theory [6,9], which we used previously as a framework for analyzing wiki activity [7]. The presented work significantly extends the scope of activity analyses and presents an evaluation with team managers. Glassman and Kang [8] propose that learning via Wikis and Web browsing is best explained as an abductive logic process, consisting of discovery and hypothesis generation, which would call for a model that reasons about prior activity to explain ongoing activity. The TeamAnalytics effort facilitates analysis of student online work contributions and how they progress over time by instructors and team managers.

TeamAnalytics also extends our prior work on workflow-based analysis of student online discussions [11,12]. We plan to make use of the computational workflow framework to support more efficient and robust approaches for assessing student online activities.

## 6. SUMMARY AND FUTURE WORK

This paper presents our initial implementation of TeamAnalytics that provides a summary of member contributions over time in Wiki space and SVN. Our initial study with team managers indicates that a summary of how individual students contribute to coding can influence how the managers evaluate and coordinate the team project. We plan to trace how the managers use the information in team coordination and assisting students. We will also explore opportunities to assist grading student teamwork with the TeamAnalytics report.

Based on the team manager comments collected so far, we plan to add more details on student coding activities including whose files were modified by whom. We are also investigating additional topic categories that can help instructors and managers track student activities. In order to receive more feedback while the team managers view the summaries, we plan to add feedback fields in the summary page so that we can capture team manager input regularly. Although the instructors do not directly manage teamwork, they can also provide input on how to make the summary more useful using such function.

Regarding the presentation of the summary results, we will follow suggestions from the instructors as well as the team managers in developing effective ways to show the summary information.

## 7. ACKNOWLEDGMENTS

The authors are indebted to USC Computer Science Professors Drs. David Wilczynski, Michael Crowley and William Cheng for their assistance. This research is supported by a grant from the National Science Foundation (Award #0941950).

## 8. REFERENCES

[1] Ben-Zvi, Dani (2007) Using Wiki to Promote Collaborative Learning in Statistics Education. In Technology Innovations in Statistics Education, vol 1, issue 1, article 4, 2007.

[2] Blei, D.M. and Ng, A.Y. and Jordan, M.I. Latent dirichlet allocation (2003). The Journal ofMachine Learning Research. Vol 3, 993—1022.

[3] Carroll, J., Neale, D., Isenhour, P., Rosson, M., Scott & McCrickard, D. (2003). Notification and awareness: synchronizing task-oriented collaborative activity. Int. J. Hum. Comput. Stud. 58(5), 605–632.

[4] Chen, H.L., Cannon, D.M., Gabrio, J., &Leifer, L. (2005, June). Using Wikis and Weblogs to Support Reflective Learning in an Introductory Engineering Design Course. Paper presented at the 2005 American Society for Engineering Education Annual Conference & Exposition.

[5] Cohen, J., (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20, 37-46.

[6] Engeström, Y. (1993). Developmental studies of work as a testbench of activity theory. In S. Chaiklin, S. and J. Lave, eds., Understanding Practice: Perspectives on Activity and Context (pp. 64–103). Cambridge: Cambridge University Press.

[7] Ganapathy, C., Shaw, E. & Kim, J. (2011) Assessing Collaborative Undergraduate Student Wikis and SVN with Technology-based Instrumentation: Relating Participation Patterns to Learning, Proceedings of the American Society of Engineering Education Conference, 2011.

[8] Glassman, M. & Kang, M. J. (2011) The logic of wikis: The possibilities of the Web 2.0 classroom. Computer-Supported Collaborative Learning (2011) 6:93–112.

[9] Kaptelinin, V., &Nardi, B. A. (2006). Acting with technology: Activity theory and interaction design. Cambridge: MIT.

[10] Moodle. http://www.moodle.org/

[11] Ma, J., Shaw, E. and Kim, J. (2011) Workflow-Based Assessment of Student Online Activities with Topic and Dialogue Role Classification, Proceedings of the AI in Education Conference, 2011.

[12] Ma, J., Shaw, E. and Kim, J. (2010) Computational Workflows for Assessing Student Learning, Proceedings of the International Conference on Intelligent Tutoring Systems Conference, 2010.

[13] Ramage, D. and Hall, D. and Nallapati, R. and Manning, C.D. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.

[14] Rick, J. & Guzdial, M. (2006) Situating CoWeb: a scholarship of application, Computer-Supported Collaborative Learning (2006) 1:89-115.

[15] Suthers, D., Dwyer, N. and Medina, R. (2010) A framework for conceptualizing, representing, and analyzing distributed interaction Vatrapu Computer-Supported Collaborative Learning (2010) 5:5–42

[16] Upton, K. and Kay, J. (2009). Proceedings of the User Modeling, Adaptation and Personalization Conference, Narcissus: Group and Individual Models to Support Small Group Work, UMAP 2010.

[17] Wang, H.-C., Lu, C.H., Yang, J.-Y., Hu, H.-W., Chious, G.-F., Chiang, Y.-T. & Hsu, W.L. (2005) An Empirical Exploration of Using Wiki in an English as a Second Language Course, In Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies.

[18] Zimmerman, A., Lorenz, A. & Oppermann, R. (2007). An operational definition of context. In: CONTEXT 2007, Lecture Notes in Artificial Intelligence n. 4635, pp. 558–6571. Springer-Verlag, Berlin.

# Automated Student Model Improvement

Kenneth R. Koedinger
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
koedinger@cmu.edu

Elizabeth A. McLaughlin
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
mimim@cs.cmu.edu

John C. Stamper
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
john@stamper.org

## ABSTRACT

Student modeling plays a critical role in developing and improving instruction and instructional technologies. We present a technique for automated improvement of student models that leverages the DataShop repository, crowd sourcing, and a version of the Learning Factors Analysis algorithm. We demonstrate this method on eleven educational technology data sets from intelligent tutors to games in a variety of domains from math to second language learning. In at least ten of the eleven cases, the method discovers improved models based on better test-set prediction in cross validation. The improvements isolate flaws in the original student models, and we show how focused investigation of flawed parts of models leads to new insights into the student learning process and suggests specific improvements for tutor design. We also discuss the great potential for future work that substitutes alternative statistical models of learning from the EDM literature or alternative model search algorithms.

## Keywords

Student modeling, model discovery, model comparison.

## 1. INTRODUCTION

Student models drive the great many instructional decisions that automated tutors currently make, whether it is how to organize instructional messages, sequence topics and problems in a curriculum, adapt pacing to student needs, or select appropriate materials and tasks. Student models also appear critical to accurate assessment of self-regulated learning skills or motivational states. A better student model yields better instruction, which leads to improved learning. More accurate skill diagnosis leads to better prediction of what a student knows which provides better assessment. Better assessment leads to more efficient learning overall.

Cognitive Task Analysis (CTA) has been shown to lead to better instruction [2], and CTA is currently the best strategy for creating cognitive models. Unfortunately, CTA is an expensive and time intensive process that is fundamentally driven by human experts. The main goal of this research is to accelerate the process of improving student models. Educational data mining and machine learning techniques can be used to improve these models in an automated fashion. In this research, we combine an automated search algorithm with existing proposed student models in the DataShop repository in a crowd sourcing fashion. Using this process, we have been able to make improvements in the models that lead to important focused insights into ways to improve the instruction of the tutoring systems that use these models.

### 1.1 Background and Related Work

A number of studies have demonstrated how detailed CTA can result in dramatically better instruction [2; 7; 9]. Cognitive Task Analysis clearly works in creating cognitive models but as it currently exists, CTA has several limitations. First, CTA is more of an art than a science involving many subjective decisions. Second, the most successful CTA approaches are heavy in human effort. Approaches like structured interviews, think alouds, or developing cognitive model simulations all require high level of psychological expertise and significant time investment.

It may be possible to achieve similar outcomes using more automated techniques that utilize educational data mining and machine learning on large sets of student data. These techniques can reap many of the benefits of CTA, but with less effort and expertise than currently required. CTA typically produces a symbolic representation of a student model, for instance, a rule-based production system of the skills in a domain. An alternative is to use data and statistical inference to create a student model involving continuous parameters over latent variables with links to observed student performance variables.

In domains where cognitive models have been created, the learning curves derived from these models are a source for data-driven model revision. Others have applied learning curve analysis in the improvement of tutors. In one study of a constraint-based tutor for teaching database programming [10], learning curve analysis was used to analyze log data and make student model improvements. Creating models of student performance is useful both for student assessment and for better student instruction. With respect to assessment, use of such models in on-line systems might in fact be just as good at the job that standardized tests are intended to perform [5]. With respect to better instruction, such models are the basis for the kind of student-customized adaptive instruction that intelligent tutoring systems can provide [6]. Cognitive Tutors for mathematics are now in use in more than 2600 schools across the US for some 600,000 students per year. While these systems have been quite successful, there is room for improvement in the student models that drive their behavior.

The DataShop repository at the Pittsburgh Science of Learning Center (http://learnlab.org/datashop) provides a resource for educators and researchers to create, modify, and evaluate student models [8]. DataShop is an open data repository for educational data with associated visualization and analysis tools. DataShop has data from thousands of students derived from interactions with on-line course materials and intelligent tutoring systems. The data is fine-grained, with student actions recorded roughly every 20 seconds, and it is longitudinal, spanning semester or yearlong courses. As of December 2011, over 300 datasets are stored including over 70 million student actions, which equates to over 190,000 student hours of data. Most student actions are "coded" meaning they are not only graded as correct or incorrect, but are categorized in terms of the hypothesized competencies or "Knowledge Components" needed to perform that action.

In DataShop terminology, Knowledge Components (KCs) are used to represent pieces of knowledge, concepts or skills that students need to solve problems. When a specific set of KCs are

mapped to a set of instructional tasks (usually steps in problems) they form a KC Model. A KC model is a specific kind of student model. DataShop provides an easy interface for exporting, modifying, and importing KC models, allowing researchers to evaluate and compare alternative KC models.

## 1.2 Student Model Improvement

A number of automated and hand search methods of exploring the cognitive model space have been proposed [1; 4; 10; 11; 13; 14]. These approaches create alternative models that are scored against existing models using one of several metrics for model prediction of student performance and how it changes over time. These include Akaike information criterion (AIC), Bayesian information criterion (BIC), and cross validation [12]. A statistical model is needed to make predictions about changes in student performance and DataShop uses an extension of item response theory that incorporates a growth or learning term [cf., 3; 14]. We refer to this model as the "Additive Factors Model" (AFM) [1; 13] and it is shown in Figure 1. In this statistical model, the discrete portion of the student model is represented by $q_{jk}$, the so-called "Q matrix" [16], which maps hypothesized difficulty or learning factors (the knowledge components or skills) to steps in problems. These factors are hypothesized causes for difficulty ($\beta_k$) or for learning improvement as students practice ($\gamma_k$). AFM gives a probability that a student i will get a problem step j correct based on the student's baseline proficiency ($\theta_i$), the baseline difficulty ($\beta_k$) of the required KCs ($q_{jk}$), and the improvement ($\gamma_k$) in those KCs as the student gets practice opportunities ($T_{ik}$).

$$\ln\left(\frac{p_{ij}}{1-p_{ij}}\right) = \theta_i + \sum_{k=1}^{K} q_{jk}\beta_k + \sum_{k=1}^{K} q_{jk}\gamma_k T_{ik}$$

**Figure 1.** In the Additive Factors Model (AFM), the probability student i gets step j correct ($p_{ij}$) is proportional to the overall proficiency of student i ($\theta_i$) plus for each factor or knowledge component k present for this step j (indicated by $q_{jk}$), add the base difficulty of that factor ($\beta_k$) and the product of the number of practice opportunities this student (i) has had to learn this factor ($T_{ik}$) and the amount gained for each opportunity ($\gamma_k$).

Previous efforts to evaluate cognitive models have used BIC as the evaluation criteria [1]. BIC reduces the chances of over-fitting the data by penalizing for increasing the number of parameters in the model. It is much faster to compute than cross validation and reasonably predicts the results of cross validation. When time is not an issue, cross validation is preferred. There is currently no consensus on how to perform the folding process in cross validation for student model comparison and we discuss three alternatives below (which are in use in DataShop).

# 2. THE CREATION AND EVALUATION OF STUDENT MODELS IN DATASHOP

Before discussing automated generation of student models, we first describe how DataShop supports researchers in creating and evaluating alternative knowledge component-based student models (represented as Q matrices). The log data collected in DataShop is composed of student attempts on problem steps in a given set of instruction. Each of these problem steps can be tied to one or more skills or knowledge components. This linking of problem steps to knowledge components is called a KC model in DataShop and represents a student model for that set of instruction. Researchers can export KC models from DataShop, modify them using Excel or another editor, and then import a new model into DataShop for comparison.

KC models in DataShop are fit to data using the AFM equation in Figure 1, and metrics for AIC, BIC, and three versions of cross validation are provided to evaluate and compare different models.

We illustrate the modification of a KC model to produce an improved model with implications for tutor redesign. The example data comes from a data set called Geometry9697 and can be found in the DataShop repository under Public Datasets. Figure 2 shows a screen shot of (a more recent version of) the tutor used in generating the data.

In this example, the best hand-generated model divides the ALT:COMPOSE-BY-ADDITION KC of the original in-use model into three KCs: *Subtract, compose-by-addition*, and *decompose*. The original ALT:COMPOSE-BY-ADDITION KC labels steps where the student must find the area of an irregular shape that may be the sum or difference of two regular shapes (e.g., what's left when a circle is cut from a square). This KC was targeted for improvement because, as shown in the top of Figure 3, it was found to have a non-smooth learning curve (a large difference between actual and predicted values) and although it is relatively difficult (26% error rate), the learning curve does not indicate any learning (the error rate does not go down with opportunities) and, correspondingly, the AFM slope estimate ($\gamma j$) is zero. As described in [15], these features of a learning curve (not smooth, not low, and not declining) are indicators of a poorly defined KC. A KC may be improved by investigating the problem steps it labels, usually focusing on those where the error rate is much higher (or lower) than normal as in opportunities 12, 15-18, etc., shown in the curve at the top of Figure 3. The analyst seeks a feature of these problem steps that may change the difficulty of performing or learning that step (i.e., a difficulty or learning factor) that is not shared by the other problem steps. In this case, almost all of the hardest problem steps required students to identify the two regular shapes that make up a target irregular shape (i.e., to visually "decompose" and set subgoals to find the area of these regular shapes first). These problem steps were relabeled with a KC called *decompose*. In other problems, the ALT:COMPOSE-BY-ADDITION steps came after an explicit scaffold given to students to find the area of the regular shapes (e.g., a prompt to find the square and circle areas before finding the leftover). These remaining problem steps were relabeled *compose-by-addition*. We say that ALT:COMPOSE-BY-ADDITION was "split" by the decompose factor to produce a new "decompose" KC and a modified "compose-by-addition" KC with fewer steps associated with it.

By inspecting a subset of particularly easy problem steps, another factor was identified (repeated steps in the same problem) and these steps were labeled *Subtract*. That is, ALT:COMPOSE-BY-ADDITION was further split into a third set of steps. In sum, the new KC model splits the ALT:COMPOSE-BY-ADDITION KC in the Original model (which labels 20 steps) into three different KCs: *compose-by-addition* (6 steps), *decompose* (8 steps), and *Subtract* (6 steps).

The bottom of Figure 3 shows, for all three new KCs, the resulting learning curves and the parameter estimates for the difficulty intercepts (in both logit terms, $\beta_k$, and converted to a probability) and for the KC learning slopes ($\gamma_k$). Inspecting the empirical learning curves (red lines), all three look smoother than the original ALT:COMPOSE-BY-ADDITION, thus meeting the

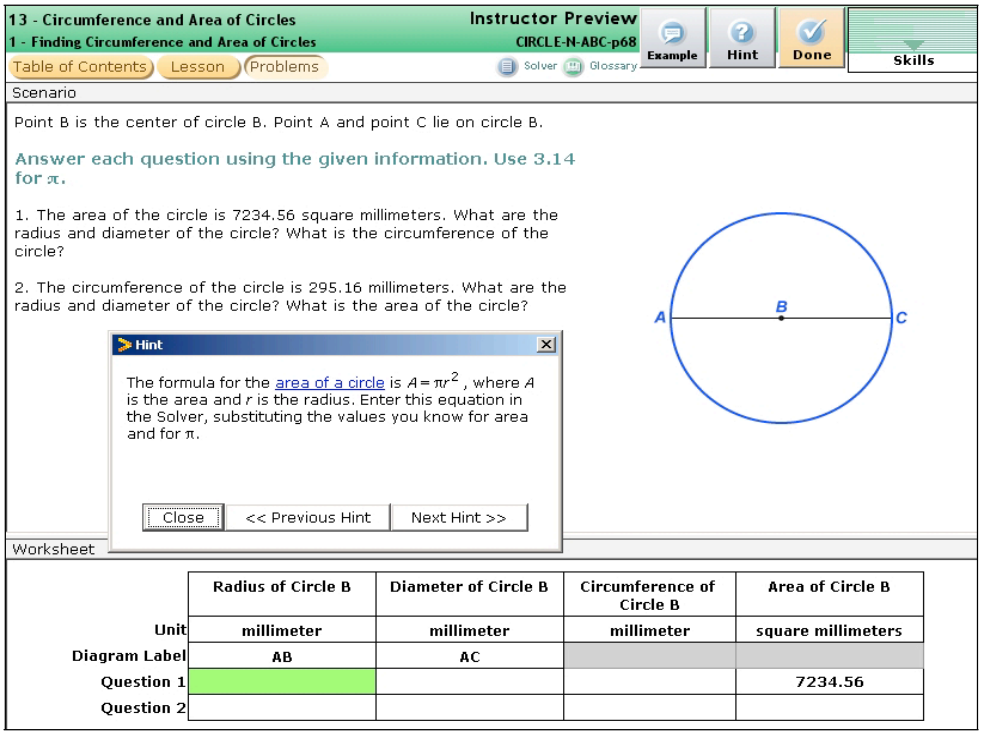



**Figure 2.** A screen shot of the problem Circle N present in the area unit of the Geometry Cognitive Tutor. In Question 1, students are given the area of the circle and must find the radius, diameter, and circumference. In Question 2, students are given the circumference and must find the radius, diameter, and area. The Hint message is in response to a student hint request for the highlighted cell.

"smoothness" criteria. The *decompose* KC learning curve appears to be declining in error rate and, correspondingly, the slope parameter is greater than zero indicating an improvement in the model. The *Subtract* KC curve is not declining but it is already at a low error rate from the start, so this also indicates an improvement in the model. Such a pattern indicates the KC is already known and mastered, therefore, little to no learning is expected. The story for the new *compose-by-addition* KC is mixed – while smoother, it is not declining and not already low. It might be possible to make further improvements in this KC.

The decompose KC results are quite different from the other two newly labeled KCs and the original KC (ALT:COMPOSE-BY-ADDITION) with a much higher initial error rate (57%) and a declining curve (intercept = .36, slope = .15). Given these results and the lack of mastery on the *decompose* KC after six opportunities, we recommend a higher concentration of decomposition problems with additional instructional aids such as worked examples, specific hints, and problems that isolate practice on this skill [15]. These results also indicate less practice is needed on the *Subtract* KC and corresponding problem steps could be reduced or even eliminated from the curriculum.

The statistical fit (shown in the blue lines in Figure 3) is based on the Additive Factors Model described above. Models are evaluated using AIC, BIC, and 10-fold cross validation. We report the root mean-square error (RMSE) averaged over the ten test sets in the cross validation.

For this dataset, the best models according to BIC and cross validation are ones that incorporate the distinction between unscaffolded (decompose) and scaffolded problem steps. More complexity, (e.g., models with 12 and 13 KCs) pays off relative to a simpler model with 10 KCs. But more complexity is not always better. The Original production rule model in the tutor had 15 skills, yet according to BIC and cross validation measures the simpler models with 12 and 13 skills are better predictors.

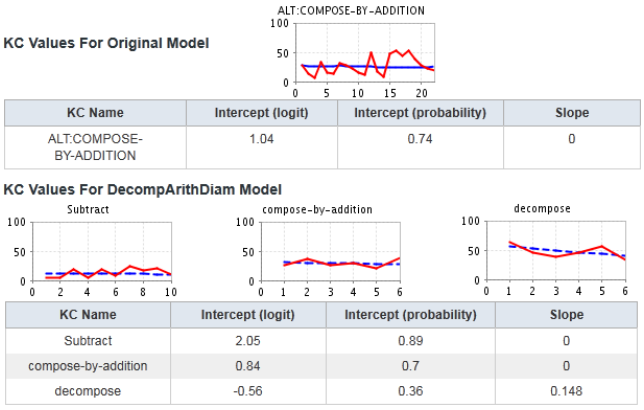



**Figure 3.** A knowledge component (KC) with a non-smooth learning curve (see top half of the figure) is replaced in an improved student model with three new KCs with smoother curves (see bottom half of the figure).

There are a couple instructional consequences of the fact that the decompose skill was confounded with performing scaffolded decomposition and, even, with simple subtraction. First, students were able to give the appearance of mastery because they were essentially given credit for this more complex skill when they successfully performed the two simpler skills. It is possible for students to graduate on the merged skill by only getting scaffolded decomposition and subtraction steps correct and never or rarely getting an unscaffolded decompose step correct (the tutor's knowledge tracing algorithm allows for an occasional slip).

Second, because there was no differentiation of these skills, there was no way to provide any isolated or extra practice on the tough unscaffolded decompose skill. Thus, for both of these reasons, there was not enough practice of decompose.

We have illustrated how new KC models can be produced and evaluated within DataShop. This process has produced many different models across a variety of datasets. For instance, the Geometry9697 dataset had at least 10 KC models associated with it. These models range in number of KCs from 1 to 15 with a median of 12 KCs. The KC models vary on factors including the shape of a figure, the formula involved, whether a formula is applied forward or backward, whether or not this step repeats an analogous one in the same problem, etc.

We next present the use of the LFA algorithm to perform an automated search for better models. A key trick is to use, in a crowd-sourcing fashion, existing models entered by DataShop users as the basis for input to this search algorithm

# 3. APPLYING LEARNING FACTORS ANALYSIS (LFA) ACROSS DOMAINS

## 3.1 Adapting LFA for use in DataShop

Learning Factors Analysis (LFA) [1] is an algorithm that automatically finds better student models by searching through a space of KC models, represented as Q matrices, to find the one that best predicts student-learning data. The input to LFA includes a dataset of records that indicate a student, a step identifier (i.e., part of a problem or activity for which there is an observed and gradable student action), the order in which each student experiences each step, and whether the student was successful or not on the step (usually whether the students first action on a step is correct, that is, neither a hint request nor an incorrect action). The LFA input also includes a matrix, in the same form as the Q matrix described above, that indicates for each unique step (the rows) what candidate features or factors may affect student performance and learning on that step (the columns). This so-called P matrix is used, along with a set of operators, to determine the space of possible Q matrices that LFA searches over.

The output of LFA is a list of Q matrices (KC models) rank ordered (using either AIC or BIC) by how well they predict the student data. In the search process, new Q matrices are created from the current Q matrix by applying operators (split, merge, or add in the complete LFA) using a factor in the P matrix to modify some aspect of the current Q matrix.

Figure 4 provides a simple example of the search process beginning with the mapping of problem steps to Q and P matrices. In this example, a Q matrix with factors for multiplication (Mult) and subtraction (Sub) is modified by applying a split operator to a column in the Q matrix (Sub) using a column in the P matrix (Neg-result). The outcome is a new Q matrix (called Q' in Figure 4) that has the steps of Sub partitioned into two subsets (Sub-Pos and Sub-Neg) according to the values of Neg-result.

An important challenge in a broad application of LFA is determining how the P matrix gets created. In early applications of LFA, a human user (a single domain expert) created the P matrix. In the spirit of crowd sourcing, we have used the multiple hand-created KC models in DataShop as an alternative way to create the P matrix. The P matrix is automatically derived by combining all of the hypothesized KCs in the pre-existing KC models for that dataset. More specifically, the P matrix maintains the same rows (one for every step in every problem), but the columns (the hypothesized factors or KCs) are the union of all the columns in the pre-existing KC models (such that the number of columns is the sum of the number of columns across the existing models minus any duplicate columns).

| | Q | | P | | Q' split [Q,Neg Result] | | |
|---|---|---|---|---|---|---|---|
| Problem Step | Mult | Sub | Neg result | Order of Op | Mult | Sub-Pos | Sub-Neg |
| 2*8-30 => 16-30 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 16 - 30 => -14 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 30-2*8 => 20-16 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 20-16 => 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

**Figure 4.** Example of a Q matrix and P matrix mapped to problem steps and the resulting Q' matrix when Sub in the Q matrix is "split" by Neg-result from the P matrix.

LFA implements a best-first search. In each iteration of the search, the best Q matrix so far (as measured by AIC or BIC) is chosen for expansion. The LFA search process begins with the simplest possible Q-matrix where all steps involve one and only one KC. New candidate Q matrices (KC models) are generated by applying operators to the columns in the P matrix and the selected Q matrix. In the version used here, we only used the split operator: LFA creates new candidate Q matrices by splitting KCs in the current Q matrix using relevant factors in the P matrix. The search stops either after a pre-set number of iterations or when model improvements diminish (see below).

The search process results in the creation of machine-generated KC models that are usually more predictive (as measured by cross validation) than any of the starting models. Because the current implementation is only using LFA's split operator, which only generates single KC codes for each problem step, it is possible a starting model that includes multiple KCs per step could be better than any LFA model generated. Although cross validation is arguably the best way to test the predictive efficacy of a model, it is too computationally expensive to run inside the LFA search. Instead we have used BIC and, more recently, AIC as the heuristic to guide the search. After the search is complete, we test the best models using cross validation.

## 3.2 Method: Apply LFA across 11 datasets

In order to examine the LFA search process across a variety of datasets, we used eleven datasets representing five domains (geometry, algebra, fractions, English articles, and statistics) from the DataShop repository. Each dataset had from 1-16 KC models previously created by content specialists or researchers and most (65%) of these models coded a single KC per step. The number of knowledge components within models ranged from 1-48 and the number of student users ranged from 41-318. In addition to a variety of dataset characteristics, we have a wide group of researchers/authors represented: Lovett (statistics), Wiley (English articles), Booth (equation solving), Lomas (fractions), Koedinger (symbolization), Stamper, Ritter and Koedinger (geometry area).

To each dataset, we applied a version of LFA that: 1) used only the split operator, 2) started with a Q matrix with a single KC labeling all problem steps, and 3) started with a P matrix made up of the union of all existing KC models. We ran the search process twice on each dataset, once using BIC as the search heuristic and once using AIC. The search continues until a streak of 5 iterations does not produce a model with an improved heuristic value. We compared the two best models from each of the AIC and BIC runs (4 models total) with all the existing models using root mean

square error (RMSE) as determined by 10-fold cross validation. Three different cross validations were run: 1) student stratified (SSCV), 2) item stratified (ISCV), and 3) no stratification (NSCV). For student stratified and item stratified cross validation, students or items (i.e., problem steps), respectively, were randomly chosen for the folds. No stratification cross validation selects the 10 folds randomly from the dataset as a whole, irrespective of student or item. We chose item stratified cross validation as the primary metric (the one used in Tables), because we are concerned with improving tutors and item stratified cross validation corresponds most closely with a key tutor decision of what next problem to select.

## 3.3 Results: Better Models Found

Table 1 summarizes the results. Analysis of the datasets using RMSE from a 10-fold item stratified cross validation (ISCV) shows a machine-generated model is the best predictor of student performance across all eleven datasets and all four domains. This can be seen in Table 1 by noting that the RMSE values in the Best-LFA column (representing the results of machine-generated models) are all lower than those in the columns for the best model found by hand (Best-hand) and for the original model in use by the tutoring system or game (Orig-in-use).

The results from both SSCV and NSCV were mostly similar in that in 10 of 11 datasets, an LFA model was best. For the DFA-318 dataset, the Best-hand model was better on SSCV and NSCV (but not on ISCV). That model involved some steps being coded by more than one KC – such multi-KC coding is not a model that the LFA version we used (which only incorporates the split operator) can produce, however, a version with LFA's add operator could, in principle, produce such a model.

Besides reducing prediction error, the LFA-discovered models tend to better capture student learning in that the typical slopes on learning curves are steeper. The last columns in Table 1 show a comparison of the median learning slope ($\gamma_k$) for the Best-hand and Best-LFA models. In all but a couple of cases, the median learning slope is larger for the Best-LFA model than it is for the Best-hand model (e.g., 0.11 vs. 0.07 for Geometry9697).

# 4. INTERPRETING STUDENT MODEL IMPROVEMENTS

We have shown how an automated process, such as an LFA search, discovers new and improved student models. But, are these improvements substantial and interpretable? Can they be used to propose plausible improvements in a tutoring system?

One method to evaluate newly discovered machine models is to investigate when and how much each KC in the original model is changed in the transformations that produced the best-hand and best-machine models. By isolating improvement in knowledge components, areas of student difficulty can be uncovered and automated systems can be redesigned to more efficiently address student learning by focusing better instruction and more practice on more difficult skills and less practice on easier skills.

We use the Geometry9697 dataset as an example of a proposed strategy for interpreting LFA results toward model and tutor improvement. That strategy starts with inspecting the impact of model improvements on specific aspects of the original model. A key observation is that while the overall prediction error (RMSE) reductions in Table 1 may seem small, LFA is likely to make significant changes in the KC model only in a few isolated places, that is, only for some of the original KCs. Although those changes may be practically significant, they are obscured in the overall RMSE change given much of the model remains the same.

Thus, we suggest trying to identify which of the KCs in a base model are most substantially changed in the creation of a new model. One way to do so is to compute the reduction in RMSE between models for each of the KCs in the base model.

**Table 1.** The root mean square error (RMSE) for the best KC models as determined by item stratified cross validation.

| Dataset | Content area | RMSE | | | Median Learning slope (logit) | |
|---|---|---|---|---|---|---|
| | | Orig in-use | Best-hand | Best-LFA | Best-hand | Best-LFA |
| Geometry9697 | Geometry area | 0.4129 | 0.4033 | 0.4011 | 0.07 | 0.11 |
| Hampton 0506 | Geometry area | NA | 0.4022 | 0.4012 | 0.03 | 0.04 |
| Cog Discovery | Geometry area | NA | 0.3250 | 0.3244 | 0.16 | 0.16 |
| DFA-318 | Story problems | 0.4461 | 0.4407 | 0.4405 | 0.07 | 0.17 |
| DFA-318-main | Story problems | 0.4376 | 0.4287 | 0.4266 | 0.09 | 0.17 |
| Digital game | Fractions | 0.4442 | 0.4396 | 0.4346 | 0.17 | 0.14 |
| Self-explanation | Equation solving | NA | 0.4014 | 0.3927 | 0.01 | 0.04 |
| IWT 1 | English articles | 0.4262 | 0.4110 | 0.4068 | 0.10 | 0.12 |
| IWT 2 | English articles | 0.3854 | 0.3854* | 0.3806 | 0.12 | 0.16 |
| IWT 3 | English articles | 0.3970 | 0.3965 | 0.3903 | 0.05 | 0.15 |
| Statistics-Fall09 | Statistics | 0.3648 | 0.3527 | 0.3353 | ** | 0.09 |

NA: Original models (or statistics on them) were not available in some cases.
* IWT 2 dataset only has 1 model, therefore the original-in-use and best-hand models have the same RMSE.
**The best-hand model for Statistics-Fall09 dataset has only 1 KC.

More specifically, for each data point labeled by a base model KC, we find the RMSE based on the predictions of the base model and compare it with the RMSE based on the predictions of the new model. We compute the percent reduction in RMSE ((base - new)/base).

Table 2 summarizes the results of this analysis on the Geometry9697 dataset by comparing each pair of the Original, best hand, and best LFA models in terms of the Original model KCs. As anticipated, the last row shows that for most of the Original KCs (12 of 15), there is little to no improvement in the best hand and best LFA models. However, for three of the KCs, there are large reductions in prediction error. The TRIANGE-SIDE KC has the largest RMSE reduction from Original to LFA model (11.1%); however, the decrease is mostly caught in the improvements made to the best hand-model from the Original model (10.0%). The COMPOSE-BY-ADDITION KC is also improved (as described in section 2) just in the original to hand-model transition. The CIRCLE-RADIUS KC, on the other hand, realizes an almost 6% reduction of RMSE from the Original model to the best hand-model, and then another sizeable reduction of almost 4% from best-hand model to the best LFA model. This discovery of LFA represents a genuine machine-based discovery not directly anticipated by human analysts.

**Table 2.** Improvement in knowledge components in the Geometry9697 dataset measured by the percent reduction of root mean squared error (RMSE) from item-stratified cross validation.

| Original model KCs | % reduction in RMSE | | |
|---|---|---|---|
| | orig->hand | hand->LFA | orig-LFA |
| CIRCLE-RADIUS | 5.8% | 4.0% | 9.5% |
| COMPOSE-BY-ADDITION | 5.2% | 0.3% | 5.5% |
| TRIANGLE-SIDE | 10.0% | 1.2% | 11.1% |
| Range of the 12 other KCs | -.5 to 3.4% | -.3 to 1.0% | -.2 to 3.1% |

A closer look at the CIRCLE-RADIUS KC from the Original shows it is coded as three separate KCs in the best hand-model: (1) circle area, (2) circle-diam-from-given, and (3) circle-diam-from-subgoal. In all three of these KCs, computing a radius is the target skill but how it is computed depends on what component measure is provided. For example, in the Circle-N problem (see Figure 2), area is given in the first row of the table (and in the text as Question 1) and students must compute the remaining values including radius; this is labeled as the circle-area KC. In the circle-diam-from-subgoal KC, circumference is given (row 2 or Q2) and in the circle-diam-from-given KC, diameter is given. As can be seen in Figure 5, the LFA model further changes these same KCs by either combining (e.g., circle-diam-from-given combines with three other KCs to form Geometry) or splitting (e.g, circle-area splits to form a reduced version of itself and a new KC called radius-from-area). Circle-diam-from-subgoal remains the same from best hand to best machine. We use the circle-area split as an illustration of how the machine-model uncovered a useful improvement in the original model that was not anticipated by humans generating hand models.

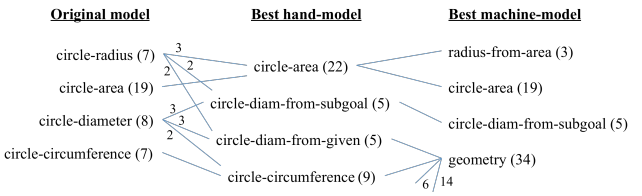



**Figure 5.** The splitting and combining of circle-radius and other related hypothesized knowledge components in going from the original-model to the best hand model to the best machine model.

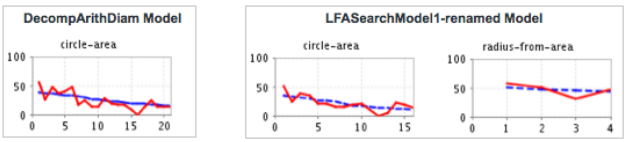
In Figure 5, we see the circle-area KC in the best hand-model has 22 problem steps but after the LFA algorithm is applied this one KC is divided into two KCs – one with 19 problem steps and the other with 3 problem steps. What is unique about these three problem steps that they split from the original 22 to form a "new" KC?

A careful examination of the three problem steps in the new radius-from-area KC reveals a backward strategy is necessary for a correct solution (e.g., finding radius when area of circle is given) unlike the other nineteen problem steps, which require a forward strategy. Although using a backward strategy is not uncommon in the dataset (about 27% of the problem steps require it), none of the other KCs were split between backward and forward by the search algorithm. In fact, eight of thirteen KCs in the best hand-model label backward steps but only circle-area benefits from being split into forward and backward versions.

Figure 6 is analogous to Figure 3 and shows the effect of the LFA discovery to split circle-area in the best hand-model (called DecompArithDiam) into circle-area (labeling fewer steps) and radius-from-area. Both resulting learning curves are smoother than the original and they have steeper slopes (.105 and .165, respectively, as compared to .068).

Performance on the circle-area backward problems (called radius-from-area) is lower (54%) than performance on the circle-area forward problems (80%), which is perhaps not a surprise. However, in this context, it is surprising that the difference in backward vs. forward performance on the other KCs is small and statistically negligible. For example, pentagon area does not split from apothem or side (backward = 66%, forward = 62%), parallelogram area does not split from parallelogram side (backward = 89%, forward = 91%), trapezoid area does not split from base or height (backward = 54%, forward = 55%), and triangle area does not split from base or height (backward = 68%, forward = 78%).

In addition to not finding a backward split for any of the area formulas for other shapes, there was no forward-backward split for other circle formulas (circumference = pi*diameter and diameter = 2*radius). A unique feature of working the circle area formula backwards is that it requires a square root operation to find the radius (after dividing the given area by pi). Thus, the uncovered knowledge component appears to be about learning when and how to employ a square root operation rather than about a general ability to apply a backward strategy, for instance, by using algebra (as suggested in the hint in Figure 2). Note, the need to employ a square root is also required in backward application of the square area formula, however, this tutor unit did not have any such problems where only the area of a square is given and the side must be found.




KC Values For DecompArithDiam Model

| KC Name | Intercept (logit) | Intercept (probability) | Slope |
|---|---|---|---|
| circle-area | 0.31 | 0.58 | 0.068 |

KC Values For LFASearchModel1-renamed Model

| KC Name | Intercept (logit) | Intercept (probability) | Slope |
|---|---|---|---|
| circle-area | 0.44 | 0.61 | 0.106 |
| radius-from-area | -0.38 | 0.41 | 0.165 |

**Figure 6.** Learning curves and model values of circle area knowledge component in best hand-model and after splitting to circle area and radius-from-area KCs in best LFA model.

The consequences for tutor changes based on this automated discovery of LFA are significant. We recommend the tutor distinguish area-to-radius problems while merging all other forward/backward area combinations (e.g., pentagon side and pentagon area). The current unit in the Geometry Tutor only has three problem steps associated with the area-to-radius KC, not enough to lead to mastery on a difficult skill. Thus, more such problems should be created. The skill tracking, which is done with skill bars in the tutor, should be modified to maintain the forward-backward distinction for the circle area formula, but merge the forward-backward distinctions for other formulas (including the other circle formulas for circumference-diameter and the diameter-radius relationships). This change should substantially reduce the time students currently take to separately (and unnecessarily) demonstrate mastery of both the forward and backward versions of each of the area formulas. This time could in turn be used to make sure that students get enough practice in cases where their prior algebra background appears insufficient for backward application of formulas, namely in use of the square root operation. Further, given the need for more square root practice, square-area-to-side problems should also be added.

## 5. CONCLUSION AND FUTURE WORK

This paper demonstrates an automated technique for the discovery of better student models using input from previously generated models. LFA discovered better student models in 11 of 11 cases as measured by item stratified cross validation and 10 of 11 cases by student stratified and non-stratified cross validation.

Although the reduction in overall error (RMSE) appears rather small, we demonstrated how this small error reduction is a consequence of most of the discovered model being essentially the same as the original. There are a few isolated changes that are statistically large and, more importantly, practically significant for tutor redesign.

A related, more nuanced point is that even in cases where there is no significant improvement in prediction error, the new model may be better in parsimony (i.e., by reducing the number of KCs). Parsimony improvements in student models are important scientifically because they simplify explanations and suggest broader transfer of learning. They are also practically important in that tutors with a more compact student model will save and focus student time.

Student models are critical to effective adaptive instruction. Different kinds of student model changes (e.g, original KCs splitting or not splitting) suggest specific tutor redesigns. A number of instructional design changes are suggested when an original KC is split into one or more new KCs. First, the skill bars and knowledge tracing need to be changed to include the new KCs and to be sure that students master all of them. Second, often a newly discovered KC will occur in too few of the existing problems and thus new problems need to be created. Third, the distinction discovered in the KC split (e.g., use of square root) may be better highlighted in the various forms of instruction that tutor may employ including worked examples, error feedback messages, and next-step hint messages. Similarly, a different set of instructional design changes are suggested when a number of original KCs are merged (i.e., are not split) into a single KC. First, the skill bars and knowledge-tracing model need to be changed to eliminate unnecessary distinctions. With corresponding knowledge tracing parameter changes, the number of practice problems a student needs to master the merged KC will be significantly reduced relative to the greater number previously needed to master each of the separate KCs. Second, the commonality discovered in the KC merge (e.g., the learning obstacle for most area operations is learning to retrieve and map the right formula, not learning how to apply it in a forward or backward direction) can be highlighted in the various forms of instruction that tutor employs. Doing so may better encourage the desired generalization and greater transfer of learning.

The general LFA algorithm includes an "add" operation that can produce multiple KC codes for a single problem step. For efficiency reasons (for larger datasets, LFA ran for multiple days) and to simplify interpretation of the results, we did not use the add operator in the version of LFA employed here. However, in the case that a new KC is added, there are instructional implications beyond those indicated above for split operations. Namely, there is the possibility of inventing new tasks that isolate a KC that might only occur with other KCs in the current set of tasks. Doing so has been demonstrated to yield significant improvements in student learning [7]. Future work should explore the use of the add operator in the LFA algorithm and, more generally, needed algorithm improvements to increase efficiency.

Some other automated techniques discover models that are difficult or impossible to understand, either toward deriving insights into student learning or making practical improvements in instruction. The output of LFA is more interpretable and convertible to tutor changes than these alternative "black box" machine learning methods that may produce Q matrices (or other latent variable representations) without consistent application of analyst-derived codes or without code labels at all. Even so, the output of LFA is complex and not trivial to interpret. Thus, we recommend a strategy (illustrated in Table 2) for isolating the practically significant student model improvements that LFA discovers. Such improvements in the student model have direct implications for many aspects of tutor design including problem development, knowledge tracing, problem selection and sequencing, skill bar display, instructional hint and hint message content. Using data to automatically improve student models and, in turn, improve instructional systems is a tremendous opportunity for educational data mining, especially as we accumulate large datasets and relevant techniques in repositories like DataShop.

A notable innovation here is a simple form of crowd sourcing of the work done by data analysts using DataShop. A significant limitation of LFA recognized in prior work [1] is the need for the P matrix input to the algorithm, that is, the human coding needed

to produce the various difficulty and learning factors on problem steps that are the basis for hypothesizing knowledge components. This limitation is addressed here by taking advantage of the DataShop facilities for creating and storing hand-built KC models and the fact that analysts have been using those facilities to create models.

Our simple version of combining human and machine intelligence toward better model discovery could be applied more broadly beyond DataShop data or student modeling. The general idea involves a web-site with these components: 1) users can modify an existing model (e.g., add new features based on feature engineering) and see whether their new model yields better predictions (e.g., DataShop's KC model leaderboard display), 2) a machine algorithm that collects features across all human-entered models (e.g., a simple union of all features), and 3) a machine algorithm that searches over the space of features to identify more predictive models (e.g., the LFA algorithm). We leave it for future research to test this proposed generalization of the approach presented here.

## ACKNOWLEDGMENTS

Our thanks to Hui Cheng for running the search algorithm, to DataShop (http://pslcdatashop.org) for dataset access, and to the National Science Foundation (#SBE-0354420) for funding of the Pittsburgh Science of Learning Center.

## 6. REFERENCES

[1] Cen, H., Koedinger, K. R., & Junker, B. (2006). Learning Factors Analysis: A general method for cognitive model evaluation and improvement. In M. Ikeda, K.D. Ashley, T.-W. Chan (Eds.) *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 164-175. Berlin: Springer-Verlag.

[2] Clark, R.E., Feldon, D., van Merriënboer, J., Yates, K., & Early, S. (2008). Cognitive task analysis. In Spector, J. M., Merrill, M.D., van Merriënboer, J., & Driscoll, M.P. (Eds.) *Handbook of research on educational communications and technology* (3rd ed.). Mahwah: Lawrence Erlbaum.

[3] Draney, K.L., Pirolli, P., & Wilson, M. (1995). A measurement model for complex cognitive skill. In P. Nichols, S.F. Chipman, & R.L. Brennan (Eds.), *Cognitively diagnostic assessment* (pp. 103–126). Hillsdale: Erlbaum.

[4] Falmange, J.C., Koppen, M., Villano, M., Doignon, J.P. & Johannesen, L. (1990). Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review*, 97, 201-224.

[5] Feng, M., Heffernan, N.T., & Koedinger, K.R. (2009). Addressing the assessment challenge in an online system that tutors as it assesses. User Modeling and User-Adapted Interaction: *The Journal of Personalization Research (UMUAI). 19*(3), pp. 243-266.

[6] Koedinger, K. R. & Aleven, V. (2007). Exploring the assistance dilemma in experiments with Cognitive Tutors. *Educational Psychology Review, 19* (3): 239-264.

[7] Koedinger, K.R. & McLaughlin, E.A. (2010). Seeing language learning inside the math: Cognitive analysis yields transfer. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. (pp. 471-476.) Austin, TX: Cognitive Science Society.

[8] Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J., (2011) A Data Repository for the EDM community: The PSLC DataShop. In Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (Eds.) *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press.

[9] Lee, R. L. (2003). Cognitive task analysis: A meta-analysis of comparative studies. Unpublished doctoral dissertation, University of Southern California, Los Angeles, California.

[10] Martin, B., & Mitrovic, A. (2004). Evaluating Intelligent Education Systems with Learning Curves in *Proceedings of the workshop on evaluation at AH2004*, Eindhoven, pp. 179-188.

[11] Martin, B., Mitrovic, T., Mathan, S., & Koedinger, K.R. (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Interaction* (2011) 21:249–283.

[12] Model Values. In PSLC DataShop. Retrieved 4/10/12 from https://pslcdatashop.web.cmu.edu/help?page=modelValues

[13] Pavlik Jr., P.I., Cen, H., Koedinger, K.R.: Learning Factors Transfer Analysis: Using Learning Curve Analysis to Automatically Generate Domain Models. In: Barnes, T., Desmarais, M., Romero, C., Ventura, S. (eds.) *Proceedings of the the 2nd International Conference on Educational Data Mining*, Cordoba, Spain, pp. 121-130 (2009).

[14] Spada, H., & McGaw, B. (1985). The assessment of learning effects with linear logistic test models. In S. E. Embretson (Ed.), *Test design: Developments in Psychology and Psychometrics*.

[15] Stamper, J. & Koedinger, K.R. (2011). Human-machine student model discovery and improvement using data. In J. Kay, S. Bull & G. Biswas (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, pp. 353-360. Berlin: Springer.

[16] Tatsuoka, K.K. (1983) Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20, 345-354.

# Automated Discovery of Speech Act Categories in Educational Games

Vasile Rus
Department of Computer Science
The University of Memphis
Memphis, TN 38152
vrus@memphis.edu

Cristian Moldovan
Department of Computer Science
The University of Memphis
Memphis, TN 38152
cmldovan@memphis.edu

Nobal Niraula
Department of Computer Science
The University of Memphis
vrus@memphis.edu
nbnraula@memphis.edu

Arthur C. Graesser
Institute for Intelligent Systems
The University of Memphis
365 Innovation Drive, Memphis, TN 38152
graesser@memphis.edu

## ABSTRACT

In this paper we address the important task of automated discovery of speech act categories in dialogue-based, multi-party educational games. Speech acts are important in dialogue-based educational systems because they help infer the student speaker's intentions (the task of speech act classification) which in turn is crucial to providing adequate feedback and scaffolding. A key step in the speech act classification task is defining the speech act categories in an underlying speech act taxonomy. Most research to date has relied on taxonomies which are guided by experts' intuitions, which we refer to as an extrinsic design of the speech act taxonomies. A pure data-driven approach would discover the natural groupings of dialogue utterances and therefore reveal the intrinsic speech act categories. To this end, this paper presents a fully-automated data-driven method to discover speech act taxonomies based on utterance clustering. Experiments were conducted on three datasets from three online educational games. This work is a step towards building speech act taxonomies based on both extrinsic (expert-driven) and intrinsic aspects (data-driven) of the target domain.

## Keywords

Speech act discovery, dialogue systems, educational games.

## 1. INTRODUCTION

An important task in dialogue-based educational systems is the detection of students' intentions from their natural language input, which we refer to as utterances. Speakers' intentions are modeled using elements from the speech act theory (Austin, 1962; Searle, 1969). Speech act theory was developed based on the "language as action" assumption as explained later. The automated detection of speaker's intentions in dialogues is known as the task of speech act classification.

Examples of speech acts are Questions, Statements, or Greetings. For instance, the hearer infers from the following utterance *How did you do that?* that the speaker is asking a Question, which informs the hearer to prepare an answer. Sometimes the speaker just states something as in the Statement, *The situation is getting worse every day.*, or greets someone as in *Hello!*.

Our work is conducted in the context of multi-party epistemic games in which chat rooms play an important role. For instance, in an Urban Science game, players take on the role of an intern for an Urban Planning company and are provided guidance from a mentor on the proper steps to be taken in redesigning a city. The players interacted with the mentor through a chat facility provided in the game. All chat among players and mentors was logged.

If the mentor role is to be automated, in a tutoring system, we need to automatically manage the dialogue which involves identifying student-players' intentions (speech act classification) based on their utterances as well as to select the best speech acts the auto-mentor system needs to produce (speech act prediction) for feedback and scaffolding. The details of the games from which we collected data are presented in the Experiments and Results section.

The task of speech act classification has been extensively addressed by the intelligent tutoring systems (ITS; [1,2]) and natural language processing (NLP; [3,4,5]) communities. The related task of speech act prediction, which is about deciding what next speech act the automated dialogue system should generate, has also been investigated to some extent [6,7,8].

The NLP and ITS communities have addressed mainly the task of speech act classification and usually in simpler setups than ours: one-to-one dialogues, e.g. between an intelligent tutor and a student user or between a ticket-booking system and a human traveler. In contrast, the present study addresses multi-party dialogues in which more than two dialogue partners are involved. This has implications on the adopted solution to classify or discover the speech acts. Some predictive features that are easy to extract in dialogues between two partners become more challenging in speech act classification or discovery for multi-party dialogues. For example, the previous speech act feature which is useful to predict the current speech act in dialogues between two partners, e.g. after a Question by one speaker an Answer by the other speaker follows, becomes more challenging in multi-party dialogues because the previous speech act is not always directly linked to the current speech act, as in the case of a third partner joining the discussion suddenly.

Furthermore, the solutions to the task of speech act classification proposed by the ITS and NLP researchers are not fully automated because the important step of specifying the speech act taxonomy is manually handled by experts [9]. The expert-generated

taxonomies are specified extrinsically as experts generate them in an ad-hoc manner without an exhaustive analysis of the available data. Indeed, Andernach, Poel, and Salomons [10] indicates that experts define taxonomies based on their intuitions with minimum information from actual data which makes it hard to define a set of rules that different human annotators (or machines) could consistently apply to data in order to derive the same speech acts for similar utterances. In general, experts define a wishful taxonomy and then the hope is the automated algorithms could learn automatically the patterns to detect the speech acts in the taxonomy. There are other lingering issues with the expert-defined taxonomies as Traum [9] pointed out. Among these issues, Traum mentions the "significant challenges for creating a taxonomy of dialogue acts that can be understood and used by researchers other than the taxonomy designers."[1] We believe that a data-driven approach to discover or at least inform the creation of speech act taxonomies could be extremely useful. This work is a step in this direction of creating taxonomies based on both extrinsic and intrinsic processes.

We propose a data-driven approach that infers the intrinsic speech act categories from the data based on the similarities of the dialogue utterances according to some model, e.g. using lexical and positional information from the utterances. The method is based on clustering algorithms, both parametric (K-Means) and non-parametric (Expectation-Maximization), to group dialogue utterances into homogeneous groups which are then used to define the speech act categories. An automated method to discover the speech categories could complement and also be used as a validation tool for expert-defined taxonomies. The natural language community has largely ignored the task of automated discovery of speech act taxonomies; there has been only one early attempt nearly two decades ago [10]. To the best of our knowledge, no previous work proposed such an automated method for speech act discovery in the area of dialogue-based intelligent tutoring systems and the larger ITS community.

Our effort fits within the grander goal of building data-driven dialogue managers [11, 12]. The closest work to our own effort in the area of educational systems is by Kristy Boyer and colleagues ([12, 13]). They automatically derived 'dialogue modes' from sequences of dialogue acts (a modern definition of speech acts), instead of asking experts to define the dialogue modes. The best number of dialogue modes is found intrinsically by selecting inferred sets of dialogue modes that maximize a log-likelihood fit function. We follow a similar idea but instead of inferring sets of dialogue modes we infer categories of speech acts and rely on clustering algorithms instead of Hidden Markov Models as Boyer and colleagues did. Hidden Markov Models are best suited for inferring hidden variables from sequences of events. In our case, we were interested in the discovery of hidden similarity patterns among individual utterances and thus clustering was a natural choice. We chose K-Means and Expectation Maximization (EM) as the clustering algorithms. The former requires a priori specification of the number of clusters expected while EM can automatically infer the number of clusters through cross validation. The appealing of K-Means is its simplicity and the ease of interpretation, e.g. a centroid vector for each cluster is provided which can be used to interpret the cluster. In the case of K-Means we experimented with several pre-specified numbers of clusters. By default, the results thus obtained are compared with the expert-defined number of clusters, i.e. the expert speech act categories.

The rest of the paper is organized as in the followings. The next section provides an overview of speech act theory and speech act taxonomy work. We then provide the conceptual framework behind our basic idea to cluster dialogue utterances. The Experiments and Results section describes our experimental setup and the results obtained. We conclude with Conclusions and Future Work.

## 2. RELATED WORK

Speech act theory has been developed based on the language as action assumption which states that when people say something they do something. Speech act is a construct in linguistics and the philosophy of language that refers to the way natural language performs actions in human-to-human language interactions, such as dialogues. Its contemporary use goes back to John L. Austin's theory of locutionary, illocutionary and perlocutionary acts [14]. According to Searle [15], there are three levels of action carried by language in parallel. First, there is the locutionary act which consists of the actual utterance and its exterior meaning. Second, there is the illocutionary act, which is the real intended meaning of the utterance, its semantic force. Third, there is the perlocutionary act which is the practical effect of the utterance, such as scaring, persuading, and encouraging.

It is interesting to notice that the locutionary act is a feature of any kind of language, not only natural ones, and that it does not depend on the existence of any actor. In contrast, an illocutionary act needs the existence of an environment outside language and an actor that possesses intentions, in other words an entity that uses language for acting in the outside environment. Finally, a perlocutionary act needs the belief of the first agent in the existence of a second entity and the possibility of a successful communication attempt: the effect of language on the second entity, whether the intended one or not, is taking place in the environment outside language, for which language exists as a communication medium. As opposed to the locutionary act, the illocutionary and perlocutionary acts do not exist in purely descriptive languages (like chemical formulas), nor in languages built mainly for functional purposes (like programming languages). They are an indispensable feature of natural language but they are also present in languages built for communication purposes, like the languages of signs or the conventions of warning signals.

In a few words, the locutionary act is the act of saying something, the illocutionary act is an act performed in saying something, and the perlocutionary act is an act performed by saying something. For example, the phrase "Don't go into the water" might be interpreted at the three act levels in the following way: the locutionary level is the utterance itself, the morphologically and syntactically correct usage of a sequence of words; the illocutionary level is the act of warning about the possible dangers of going into the water; finally, the perlocutionary level is the actual persuasion, if any, performed on the hearers of the message, to not go into the water.

[1] Dialogue acts, speech acts, communicative acts, conversational acts, conversational moves, or dialogue moves are terms used by different researchers to refer to the same general concept [9].

| Speech Act Category | Example | Count |
|---|---|---|
| **Statement** | I'll be your planning consultant. | 605 |
| **Request** | Click that and click "New Staff Page" | 343 |
| **Reaction** | Ah, I see. | 642 |
| **MetaStatement** | i didn't understand what maya wanted | 176 |
| **Greeting** | Hello! | 103 |
| **ExpressiveEvaluation** | good!!!!!!!!!! | 166 |
| **Question** | why am i getting notes from people not in my group? | 646 |
| **Other** | same thing what | 87 |

**Table 1.** Our flat Speech Act Taxonomy with examples for each speech act category.

The notion of speech act is closely linked to the illocutionary level of language. The idea of an illocutionary act can be best captured by emphasizing that ”by saying something, we do something” [14]. Usual illocutionary acts are: greeting (”Hello, John!”), describing (”It’s snowing.”), asking questions (”Is it snowing?”), making requests (”Could you pass the salt?”), giving an order (”Drop your weapon!”), making a warning (”The floor is wet!”), or making a promise (”I’ll return it on time.”). The illocutionary force is not always obvious and could also be composed of different components. As an example, the phrase ”It’s cold in this room!” might be interpreted as having the intention of simply describing the room, or criticizing someone for not keeping the room warm, or requesting someone to close the window, or a combination of the above.

A speech act could be described as the sum of the illocutionary forces carried by an utterance. It is worth mentioning that within one utterance, speech acts can be hierarchical, hence the existence of a division between direct and indirect speech acts, the latter being those by which one says more than what is literally said, in other words, the deeper level of intentional meaning. In the phrase, “Would you mind passing me the salt?”, the direct speech act is the request best described by “Are you willing to do that for me?” while the indirect speech act is the request “I need you to give me the salt.” In a similar way, in the phrase “Bill and Wendy lost a lot of weight with a diet and daily exercise.” the direct speech act is the actual statement of what happened “They did this by doing that.”, while the indirect speech act could be the encouraging “If you do the same, you could lose a lot of weight too.”

The present study assumes there is one speech act per utterance and the set of speech acts used are all at the same level of depth thereby forming a flat hierarchy. These simplification assumptions are appropriate for a first attempt at automating the speech act discovery process.

## 2.1 Speech Act Taxonomies

As already mentioned, the tasks of speech act classification and prediction requires the existence of a predefined set of speech act categories or speech act taxonomy.

Researchers agree that defining a taxonomy in general and a speech act taxonomy in particular implies a balancing act between power and simplicity ([9, 16]). That is, defining a taxonomy implies interactions between the experts’ conceptual view of the target domain with an emphasis on power, i.e. capturing fine distinctions that would maximize reaching the goal the taxonomy will serve such as effective tutoring dialogue in our case, and the need for reliable annotation and predictions, i.e. maximizing the reliability with which human annotators can tag the speech acts in which case a few, well-defined categories are better than many, sophisticated categories.

Less emphasis has been paid to the relation between the taxonomy and the actual method to automatically recognize the speech acts in the taxonomy. In other words, taxonomies were refined by observing how reliably human annotators can use them to annotate data D’Andrade and Wish [17]. The degree to which the human annotators’ process may be replicated through an automated method or the intrinsic similarities among dialogue utterances within the constraints of a chosen model, e.g. leading tokens utterances [18], has been left as an afterthought. Our work is a step towards building taxonomies based on both expert and data-driven approaches which we believe could lead to a needed trade-off between power and accuracy. That is, while expert-defined taxonomies could lead to best outcomes conceptually but may sometimes be hard to detect, the data-driven approaches would lead to taxonomies that are derived from patterns in the data and would therefore result in good speech act classification performance. A mixed approach could provide the necessary trade-off between desirable speech act categories and classification performance. It should be noted that experts do consult data, in a limited way, when deriving their taxonomies [17]. However, an automated method for grouping dialogue utterances as proposed here would infer speech act categories from the entire available data in a systematic way.

We analyzed the speech act taxonomies proposed by researchers over the years. Some are flat while others are multi-layered. The layers in the multi-layered taxonomies can be viewed as levels, in which higher level speech acts are composed of lower level speech acts, or ranks, in which layers represent different phenomena [9]. We present next a summary of the most important ones as judged from a history and relevance to our own work.

The classic categorization of Austin [14] postulates five major speech act classes based on five categories of performative verbs: Expositives - verbs asserting or expounding views, classifying usages and references; Exercitives – verbs issuing a decision that something is to be so, as distinct from a judgement that it is so; Verdictives - verbs delivering a finding, official or unofficial, upon evidence or reason as to value or fact; Commissives - verbs commiting the speaker to some course of action; and Behabitives - verbs involving the attitudinal reaction of the speaker to someone’s conduct or fortunes [17].

The taxonomy proposed by Searle [15] consists of six major classes: Representatives - committing the speaker to something’s being the case; Directives - attempt by speaker to get the hearer to do something; Commissives – committing the speaker to some course of action; Expressives - expressing the psychological state specified; Declarations - bringing into existence the state

described in the proposition and Representative; and Declarations - giving an authoritative decision about some fact.

The category scheme proposed by D'Andrade and Wish [17] treats most utterances as conveying more than one speech act and does not attempt to establish a hierarchical order among multiple speech acts. The primary motivation for the speech act coding system was a desire to investigate correspondences between speech acts and adjectival "dimensions" descriptive of interpersonal behavior. In order for a classifying system to be useful for measuring interpersonal communication, the distinctions reflected by the coding scheme should be relevant to native speakers' perceptions and evaluations of interaction. Their classes are: Assertions (Expositives), Questions (Interrogatives), Requests and Directives (Exercitives), Reactions, Expressive Evaluations (Behabitives), Commitments (Commissives) and Declarations (Verdictives, Operatives).

While there seems to be some consensus on the existence of some speech acts, like greetings, questions, answers, etc., the efficiency of a particular taxonomy for solving a particular problem ultimately rests on the task at hand. For instance, Olney and colleagues [19] used a taxonomy that divided questions into 16 subcategories and had only 3 classes for the rest of the utterances, which was suitable for a particular intelligent tutoring environment. The 16 subclasses of Questions were: Verification, Disjunctive, Concept Completion, Feature Specification, Quantification, Definition, Example, Comparison, Interpretation, Causal Antecedent, Causal Consequence, Goal Orientation, Instrumental/Procedural, Enablement, Expectational and Judgmental.

In the case of Verbmobil, a research project aiming to develop a system that can recognize, translate and produce natural utterances, the taxonomy used takes into consideration in which of the five dialogue phases the actual speech acts occur. The main classes of their taxonomy tree are: Request, Suggest, Convention, Inform and Feedback which all yield subclasses. For instance, the Convention class is composed of the following subclasses: Thank, Deliberate, Introduce, Politeness Formula and Greeting [20].

In our work, we will use the set of speech act categories, shown in Table 1. The speech act categories are based on theoretical schemes that also can be reliably coded by trained judges [14, 15, 17, 19]. We use this reference taxonomy as a benchmark for comparison purposes with the automatically derived set of speech act categories.

## 3. THE APPROACH

Our approach to the automatic identification of speech acts classes is achieved using clustering algorithms.

Clustering is the unsupervised classification of data points (usually represented as vectors in a multidimensional space) into groups (clusters) based on similarity. A cluster is therefore a collection of objects which are similar to each other in the same cluster and are dissimilar to objects belonging to other clusters. The clustering problem has been addressed in many contexts and by researchers in many disciplines. This reflects the broad appeal of clustering and its usefulness as one of the steps in exploratory data analysis. In our case, we use clustering to discover intrinsic speech acts in dialogues from online educational games.

Table 2 offers examples of utterances belonging to three different speech act categories as defined by experts. In our method, the clustering algorithm would be fed a set of utterances of this type (see Table 2) and produce clusters in which similar utterances, i.e. utterances encoding the same speech act, belong to the same cluster. A quick post-hoc analysis by a human interpreter of the clusters thus obtained would allow the labeling of each cluster with a speech act label. For instance, by analyzing the utterances in the first column in Table 2, we immediately realize that they are all greetings and therefore a good label for such a cluster would be Greetings corresponding to the speech act category of Greetings. In this paper, however, we use the expert-labeled speech act categories to evaluate the obtained clusters.

An important step in clustering a set of data points, including dialogue data, is how to represent the data. In general, clustering algorithms require a vector representation. The dimensionality of the vector space is a choice the experimenter makes. In our case of clustering dialogue utterances, we rely on the hypothesis that good speakers in collaborative (as opposed to competitive or deceitful) dialogues make their intentions clear early on in their utterances allowing hearers to detect the speakers' intentions. Intuitively, the first few words of a dialog utterance are very informative of that utterances speech act. We could even show that some categories follow certain patterns. For instance, Questions usually begin with a wh- word while speech acts such as Greetings use a relatively small bag of words and expressions, i.e. Greetings are closed-class of utterances similar to function words such as prepositions or determiners.

In the case of other classes, distinguishing the speech act after just the first few words is not trivial, but possible. It should be noted that in typed dialogue, which is a variation of spoken dialogue, some information is lost. For instance, humans use spoken indicators such as the intonation to identify the speech act of a spoken utterance. We must also recognize that the indicators allowing humans to classify speech acts also include the expectations created by previous speech acts, which are discourse patterns learned naturally. For instance, after a first greeting another greeting that replies to the first one is more likely. In multi-party dialogue the previous speech act is more complex so consecutive utterances may or may not be directly related. We ignored such intonation and contextual clues so far in our work in order to explore the potential of classifying speech acts based on words alone. We do plan to incorporate contextual clues in future experiments.

One other argument in favor of this leading words assumption is the evidence that hearers start responding immediately (within milliseconds) or sometimes before speakers finish their utterances ([21] - pp.814). Further evidence of the leading words or tokens hypothesis has been provided by Moldovan, Rus, and Graesser [18] who showed that using "leading tokens" in an utterance leads to impressive speech act classification performance.

Therefore, we adopted a model in which each utterance is represented by its leading tokens (words and punctuation). This model includes the tokens themselves as well as their positions thus relying on lexical, punctuation, and positional information. Punctuation is useful in chat rooms as one of its functions is to encode intonational information which is lost in typed dialogues.

| Greetings | Questions | Expressive Evaluation |
|---|---|---|
| Bye | what do i say ? | nice work , Player112 . |
| Bye Player102 ! | hahah what ?? | this chat thing is sooooo cool |
| bye guys | yep what now ? | nice work everyone , check your inbox |
| Bye | what do you like to do , etc . | That 's great . |
| Bye | What sort of background qualifies you for this internship? | Player109 great . |
| Bye ! | what was in your notes ? | thanks for your help , laura |

**Table 2.** Example of dialogue utterances that belong to the same speech act category as identified by experts.

## 4. EXPERIMENTAL SETUP AND RESULTS

We present in this section the experiments we conducted and the results obtained by automatically clustering dialogue utterances in order to discover the intrinsic speech act categories in the data.

The results are reported in terms of accuracy with respect to the expert-labeled speech act categories. After clustering the utterances, the expert-assigned label of the majority of the instances in a cluster is assigned as the predicted label of the cluster and thus all the instances in that cluster are given this label. Accuracy is then computed as the percentage of correctly predicted instances.

There are two major categories of clustering algorithms. Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity. Partition based clustering algorithms identify the partition that optimizes (usually locally) a clustering criterion. Example algorithms from each category are hierarchical agglomerative (HAC) and K-means, respectively. HAC produces a hierarchical structure of clusters while K-means leads to a flat, direct clustering. In HAC, each data point is initially regarded as an individual cluster and then the task is to iteratively combine two smaller clusters into a larger one based on the distance between their data points. In the K-means algorithm, we specify a priori the number of clusters (K) we would like to have in the end. The algorithm usually starts with K seed data points which are considered as individual clusters. In subsequent iterations, the remaining data points are added to some cluster based on the distance to the centroid of each cluster. The centroid is an abstract data point of an existing cluster that is found by averaging over all the other points in the cluster. A distance metric must be defined for clustering algorithms. In our experiments, we used Euclidian and Manhattan distances. The reported results are with the Euclidian distance which produced results similar to the Manhattan distance. To perform clustering, we needed to set a couple parameters: number of clusters, which informs the clustering algorithm how many clusters to generate, and seed. The seed value is used in generating a random number which is, in turn, used for making the initial assignment of instances to clusters. In general, K-means is quite sensitive to how clusters are initially assigned and thus it is often necessary to try different values and evaluate the results. We have explored seed values between10 and 100 with an increment of 10. The best obtained results are reported, which correspond to seed values of 10 and 20. We used EM and K-means implementation from WEKA [22].

We collected dialogue utterances from three online educational games. While in general a dialogue utterance or turn may contain one or more sentences, in our context an utterance usually contains one sentence, with few exceptions. Therefore, the sentence was chosen as the unit of analysis. This choice can also be justified by the fact that it is closer to the ideal situation in which one and only one speech act is performed per unit of speech, i.e. an utterance.

A first data set used for this analysis came from a study run using an epistemic game, Urban Science. Urban Science is an educational game in which players, using iPlan, a custom-designed Geographic Information System, work as urban planners to change the look and feel of Madison, Wisconsin. They listen to people's concerns, redesign the city, and present their findings to family, friends, and planning experts. Urban Science explores how innovative technology-based learning environments modeled on the professional practices of urban planners inform students' understanding of ecology. The main goal of the game is to help players learn about ecology, develop self-confidence and presentation skills, and start to see the world through the eyes of a problem-solving urban planner.

The Urban Science chat data was collected from a November 2008 game run in Milwaukee and consists of online chat posts by the students and mentors exchanging information about the game rules, content, questions, advice, suggestions, according to the game plan. The posts, collected by the game log, were further preprocessed first by splitting them into sentences, and secondly by manually labeling each sentence with a category of the 8-class taxonomy (Statements, Requests, Reactions, Meta Statements, Greetings, Expressive Evaluations, Questions and Others).

The resulting 2768 sentenced were manually classified separately by two trained annotators. Most of the speech act categories had high levels of reliability (kappas greater than 0.7) among the human coders, but two of the categories (Meta Statement and Other) had moderate kappa scores of 0.546 to .587. The overall mean kappa score across all 8 speech act categories was 0.797.

The class distribution is shown in Table 1. If one were to randomly assign a speech act category according to these distributions, the likelihood of selecting the correct speech act category by chance would be .18. The average number of tokens per sentence is 7.57, with a Standard Deviation of 6.40.

The clustering results for the Urban Science data using the Expectation-Maximization algorithm are shown in Table 3, second column. The first column represents the number of leading tokens used for a particular trial. For instance, the third row from the top corresponds to the model in which three leading tokens

| Leading Tokens | #Clusters/Urban Science | #Clusters/Land Science | #Clusters/Nephrotex | #Clusters/Combined |
|---|---|---|---|---|
| 2 Tokens | 5/34.4% | 4/38.7% | 3/38.9% | 7/34.7% |
| **3 Tokens** | **6/40.3%** | **5/42.5%** | **4/38.2%** | **6/37.8%** |
| 4 Tokens | 4/36.2% | 5/34.2% | 4/38.7% | 6/35.4% |
| 5 Tokens | 5/39.9% | 5/36.4% | 5/36.4% | 6/37.9% |

**Table 3.** Results with Expectation-Maximization clustering algorithm.

| N | Urban Science | Land Science | Nephrotex | Combined |
|---|---|---|---|---|
| 6 clusters | 29.6% | 36.4% | 28.6% | 35.2 |
| 7 clusters | 27.2% | 31.1% | 26.9% | 31.5 |
| 8 clusters | 29.1% | 30.3% | 26.3% | 27.8 |
| 9 clusters | 31.3% | 28.9% | 26.1% | 28.0 |
| 10 clusters | 27.6% | 26.2% | 25.7% | 27.0 |

**Table 4.** Results with K-Means clustering algorithm.

were used. The results show that the three leading tokens provide the best results and yields six discovered clusters. When evaluated against expert-assigned labels, the accuracy was 40.3% for the leading three tokens model. A random guess would uniformly assign a dialogue utterance to each of the eight speech acts for an accuracy of 12.5%. Compared to the expert-defined speech act categories, the EM algorithm does not identify Greetings and Other speech acts. Greetings are mostly clustered in the predicted Reactions cluster.

Results for K-Means are shown in Table 4. The results are all for leading three tokens which was the best model when using the non-parametric EM algorithm. Remember that we do not have to specify a priori how many clusters we should expect when using the EM algorithm which is the reason we first used EM to find the best model to use for the discovery of intrinsic speech act categories in the data. The first column in Table 4 indicates the number of clusters used. We tried values around the expert-defined number of clusters, which was eight clusters.

Land Science is another computer-based educational game, in which players become interns at the office of a fictitious urban and regional planning firm. The players have to weigh the trade-offs of land use decisions in ecologically-sensitive areas, interact with virtual stakeholders, and develop land use plans for local and national sites. It is a 10 hour game played in schools or out-of-school enrichment programs.

The Land Science data was collected from the log of a game run in 2010 at Massachusetts Audubon Society. The online chat posts were split into 4131 sentences which were than manually labeled independently by two humans. The inter-judge reliability scores ranged from 0.501 for the category Other to 0.918 for the category Question, with a mean of 0.755.

The class distribution is as follows: 2.3% Others, 2.3% Expressive Evaluations, 2.7% Greetings, 7.8% Requests, 8.4% Meta Statements, 19.0% Questions, 28.2% Statements and 28.9% Reactions, which means that the chance of the corpus is .21. The average number of tokens per sentence is 6.85, with a Standard Deviation of 6.69.

The results on the Land Science data set are shown in the third column of Tables 3 and 4. The best results are again for a model in which the three leading words were used. However, in this case the number of intrinsic speech act categories, i.e. clusters, is five. MetaStatements, Greetings, and Other are not identified as clusters by the three leading tokens model and the non-parametric EM algorithm.

Nephrotex is an educational game in which undergraduate engineering students role-play as professional engineers-in-training in order to develop the skills, knowledge, identity and values of engineers. In Nephrotex, students are welcomed as early career hires into the fictitious company Nephrotex, whose core technology is the ultrafiltration unit, or dialyzer, of a hemodialysis machine. The students' assigned task is to design a next-generation dialyzer that incorporates carbon nanotubes and chemical surfactants into the hollow fibers of the dialyzer unit.

Online chat posts were collected from a game run in 2011 and subsequently split into 1000 sentences which were later manually classified by two humans. The kappa scores for each of the eight categories when comparing the two trained judges ranged from .41 for class Other to .94 for class Question with an average of .68

The class distribution shows the following hierarchy: 1.1% Others, 1.4% Greetings, 2.4% Expressive Evaluations, 4.0% Meta Statements, 5.6% Requests, 17.3% Questions, 20.2% Reactions and 48.0% Statements, which indicates that the corpus' chance is .30 The average number of tokens per sentence was 9.01, with a Standard Deviation of 6.38.

The large corpus obtained by combining the previous three corpora, consists of a number of 7899 sentences, each labeled with one of the eight speech act categories. The distribution is as follows: 2.4% Others, 2.9% Greetings, 3.6% Expressive Evaluations, 7.1% Meta Statements, 9.1% Requests, 20.3% Questions, 25.8% Reactions and 28.5% Statements, resulting in a chance of .20. The average number of tokens per sentence is 7.37, with a Standard Deviation of 6.59.

For the Nephrotex corpus, the best results are obtained using the two leading tokens. However, the results obtained with the three leading tokens are comparable in terms of accuracy but not in the number of clusters discovered, three versus four. Because the three leading tokens model has been best in the other datasets, we incline to declare it a winner in this case too.

Finally, we also experimented with a combined dataset. Results are presented in the last column of Tables 3 and 4.

## 4.1 Balanced Data Set

Because the three datasets collected were dominated by certain categories, e.g. Questions, Reactions, and Statements, we wondered about the ability of the clustering algorithms to discover the intrinsic speech act categories when the data would be uniformly distributed.

To achieve this goal, we ran experiments on a balanced dataset of speech acts by extracting from the combined data set an equal number of utterances for each speech act. In the process, we dropped the *Other* category as too few utterances were available. In the end, we obtained a balanced data set of seven speech act categories, each category containing 230 utterances each.

| N | #Clusters/Accuracy |
|---|---|
| 2 | 4/29.8% |
| 3 | 6/28.3% |
| 4 | 5/31.7% |
| 5 | 6/31.1% |

**Table 5**. Accuracy and number of clusters obtain with EM algorithm on the balanced data set.

From the results in Table 5, we can see that the accuracy is quite similar for all values of N, i.e. the number of leading words used as predicting features in clustering. The leading three words generate six clusters (out of seven in the gold standard). MetaStatements were mostly labeled as Greetings, Statements, and Expressive Evaluations. For instance, the MetaStatement, "Yay!" expressing an emotion is similar to a Greeting because of its short length and exclamation mark. For short utterances which are shorter than the number of tokens used in a given model we introduce dummy values for missing tokens, e.g. NONE. So, "Yay!" and "Hi!" have similar representations except for the first tokens which explains why they are clustered. Given that ideally we would like to have a trade-off between the complexity of the model used, in our case defined by how many tokens are employed (the more tokens the more complex the model), discrimination power (number of distinguishable clusters), and performance, we conclude from the results in Table 5 that using the three leading words is best.

# 5. CONCLUSIONS AND FUTURE WORK

We proposed in this paper a fully automated method to speech act discovery. As we already mentioned, this work is a step towards a process of defining the speech act taxonomy using both extrinsic and intrinsic aspects of the target domain. The extrinsic aspects comprise of the goals of the system that needs the speech act taxonomy and the experts' knowledge and biases. The intrinsic aspects relate to the actual similarities of the actual data. A trade-off between the extrinsic and intrinsic forces could lead to a robust speech act taxonomy that is both informed by experts' views and by the actual data.

We presented results on the original dataset as well as on balanced datasets in which the gold standard (i.e., the speech act categories are validated by experts) had same numbers of utterances for each speech act. The balanced datasets offer a more fair comparison of the clustering method of the utterances in our epistemic games. However, sometimes domains such as educational systems may be biased towards particular speech acts in which case the original datasets offers us a view at the "real" world and how the proposed methods work in real settings.

A drawback of the proposed model for representing dialogue utterances, i.e. the N leading tokens, is that the distance between two dialogue utterances is based on string operations rather than lexico-semantic distances which would be more meaningful for natural language dialogues. That is, two utterances that contain the words 'hi' and 'hi' would be close in a string-based representation while 'hi' and 'hello' or 'hi' and 'bye' would not. While for the former example of 'hi' and 'bye' one could argue for the creation of a different cluster, or speech act category, for the former they should definitely be in the same cluster. One solution is to modify the clustering library in WEKA [22] to include a lexico-semantic distance based on word-to-word similarity measures, e.g. using the WordNet similarity library [23]. We do plan to explore this line of research in the future.

As one last conclusion, our work showed that there is close relationship between the model used, e.g. the number of leading tokens, and the number of intrinsic clusters found in the data. This result should inform the developers of speech act classifier who used a particular model about the power of that model to discover the intrinsic, extrinsic, or intrinsic-extrinsic speech act categories adopted.

# 6. ACKNOWLEDGMENTS

This research was supported in part by Institute for Education Sciences under awards R305A100875 and by the National Science Foundation awards #0904909 and NSF#0918409. Any opinions, findings, and conclusions or recommendations expressed in this material are solely the authors' and do not necessarily reflect the views of the sponsoring agencies.

# 7. REFERENCES

[1] Marineau, J., Wiemer-Hastings, P., Harter, D., Olde, B., Chipman, P., Karnavat, A., Pomeroy, V., Graesser, A., and the TRG. (2000). Classification of speech acts in tutorial dialog. In Proceedings of the workshop on modeling human teaching tactics and strategies at the Intelligent Tutoring Systems 2000 conference, pp. 65–71, 2000.

[2] Serafin, R. and Di Eugenio, B. (2004). FLSA: Extending Latent Semantic Analysis with features for dialogue act classification. ACL04, 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain.

[3] Reithinger, N. and Maier. E. (1995). Utilizing statistical dialogue act processing in Verbmobil. In ACL95, Proceedings of the 33rd AnnualMeeting of the Association for Computational Linguistics.

[4] Ries, K. (1999). HMM and Neural Network Based Speech Act Detection. In Proceedings of ICASSP 99.

[5] Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C., and Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. Computational Linguistics, 26(3):339–373.

[6] Nagata, M. and Morimoto, T. (1993). An experimental statistical dialogue model to predict the Speech Act Type of

the next utterance. In Proceedings of the International Symposium on Spoken Dialogue (ISSD-93), pages 83--86, Waseda University, Tokyo, Japan.

[7] Reithinger, N. (1995). Some Experiments in Speech Act Prediction. In AAAI 95 Spring Symposium on Empirical Methods in Discourse Interpretation and Generation, Stanford University.

[8] Bangalore, S., and Stent, A. (2009). Incremental parsing models for dialog task structure. In Proceedings of the Meeting of the European Chapter of the Association for Computational Linguistics.

[9] Traum, D. R. (2000). 20 Questions for Dialogue Act Taxonomies, in Journal of Semantics, 17(1):7--30, 2000.

[10] Andernach, T., Poel, M. and Salomons, E. (1997). Finding classes of dialogue utterances with Kohonen networks. In: Daelemans, W., van den Bosch, A. and Weijters, A., editors, Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing Tasks, pp. 85-94, Prague, Czech Republic.

[11] Bangalore, S., Di Fabbrizio, G., & Stent, A. (2008). Learning the structure of task-driven human-human dialogs. IEEE Transactions on Audio, Speech, and Language Processing, 16(7), 1249-1259.

[12] Boyer, K.E., Ha, E.Y., Phillips, R., Wallis, M. D., Vouk, M.A. and Lester, J.C. (2010). Dialogue Act Modeling in a Complex Task-Oriented Domain. In Proceedings of the 11th Annual SIGDIAL Meeting on Discourse and Dialogue, Tokyo, Japan, 2010, 297-305.

[13] Boyer, K.E., Ha, E.Y., Wallis, M.D., Phillips, R., Vouk, M.A., and Lester, J.C. (2009). Discovering Tutorial Dialogue Strategies with Hidden Markov Models. (2009). In Proceedings of the 14th International Conference on Artificial Intelligence in Education (AIED '09), Brighton, U.K., 2009, 141-148.

[14] Austin, J.L. 1962. How to do Things with Words. Oxford University Press, 1962.

[15] Searle, J.R. 1969. Speech Acts. Cambridge University Press, GB, 1969.

[16] Nielsen, R.D., Buckingham, J., Knoll, G., Marsh, B., and Palen, L. (2008). A Taxonomy of questions for question generation. In Vasile Rus and Art Graesser (Eds.): Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge, Arlington, Virginia, September 25-26, 2008.

[17] D'Andrade, R.G.; and Wish, M. (1985). Speech Act Theory in Quantitative Research on Interpersonal Behavior. Discourse Processes 8:2:229-258, 1985.

[18] Moldovan, C., Rus, V., & Graesser, A.C. (2011). Automated Speech Act Classification for Online Chat, The 22nd Midwest Artificial Intelligence and Cognitive Science Conference, Cincinatti, OH, April 2011.

[19] Olney, A.; Louwerse, M.; Mathews, E.; Marineau, J.; Hite-Mitchell, H.; and Graesser, A. (2003). Utterance Classification in AutoTutor. Building Educational Applications using Natural Language Processing: Proceedings of the Human Language Technology, Philadelphia, PA.

[20] Alexandersson, J.; Buschbeck-Wolf, B.; Fujinami, T.; Maier, E.; Reithinger, N.; Schmitz, B.; Siegel, M. (1998). Dialogue Acts in VerbMobil-2. volume 226, Verbmobil Report, German Research Center for Artificial Intelligence (DFKI), Saarbrcken, 1998.

[21] Jurafsky, D. and Martin, J.H.( 2009). Speech and Language Processing. Prentice Hall, 2009.

[22] Witten, I. H.; and Frank, E. (2005). Data Mining: Practical machine learning tools and techniques, 2nd Edition. Morgan Kaufmann, San Francisco.

[23] Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). WordNet::Similarity - Measuring the Relatedness of Concepts, In the Proceedings of the Nineteenth Na-tional Conference on Artificial Intelligence (AAAI-04), pp. 1024-1025, July 25-29, 2004, San Jose, CA (Systems Demo).

## About the authors:

**Vasile Rus** is currently and Associate Professor of Computer Science and Institute for Intelligent Systems at The University of Memphis. His research interests are in the areas of language and information processing with applications to intelligent tutoring systems and knowledge management in software development. Dr. Rus is the Principal Investigator of the state-of-the-art intelligenty tutoring system DeepTutor – see www.deeptutor.org.

**Cristian Moldovan** is currently a PhD student in Computer Science at the University of Memphis. His main area of interest is Natural Language Processing with applications in Intelligent Tutoring Systems, Dialogue Processing, and Question Generation. He is currently working as a Research Assistant for the DeepTutor project, a state-of-the-art Intelligent Tutoring System based on learning progressions and deep language processing methods, and for AutoMentor, which aims to create an Intelligent Autonomous Mentor for online-chat dialogue-based educational games.

**Nobal Niraula** received the B.E. in computer engineering from Pulchowk Campus, Tribhuvan University, Nepal in 2005, the M.E. in information and communication technology and the M.Sc. in communication networks and services in 2008 from Asian Institute of Technology, Thailand and Telecom SudParis, France respectively. He is currently pursuing PhD studies at The University of Memphis, USA, where he works on the DeepTutor project. His research interests are primarily in Natural Language Processing, Machine Learning, Data Mining, Semantic Web, P2P and Ad hoc networks.

**Arthur C. Graesser** is a professor in the Department of Psychology and the Institute of Intelligent Systems at the University of Memphis and is a Senior Research Fellow in the Department of Education at the University of Oxford. His primary research interests are in cognitive science, discourse processing, and the learning sciences. He served as editor of the journal Discourse Processes (1996–2005) and Journal of Educational Psychology (2009-2014) and as presidents of the Society for Text and Discourse (2007-2010) and the International Society for Artificial Intelligence in Education (2007-2009). In 2011 he received a major award from American Psychological Association on Distinguished Contributions of Applications of Psychology to Education and Training.

# Co-Clustering by Bipartite Spectral Graph Partitioning for Out-of-Tutor Prediction

Shubhendu Trivedi
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
shubhendu_trivedi@ieee.org

Zachary A. Pardos
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
zpardos@wpi.edu

Gábor N. Sárközy
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
gsarkozy@cs.wpi.edu

Neil T. Heffernan
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
nth@cs.wpi.edu

## ABSTRACT

Learning a more distributed representation of the input feature space is a powerful method to boost the performance of a given predictor. Often this is accomplished by partitioning the data into homogeneous groups by clustering so that separate models could be trained on each cluster. Intuitively each such predictor is a better representative of the members of the given cluster than a predictor trained on the entire data-set. Previous work has used this basic premise to construct a simple yet strong bagging strategy. However, such models have one significant drawback: Instances (such as students) are clustered while features (tutor usage features/items) are left alone. One-way clustering by using some objective function measures the degree of homogeneity between data instances. Often it is noticed that features also influence final prediction in homogeneous groups. This indicates a duality in the relationship between clusters of instances and clusters of features. Co-Clustering simultaneously measures the degree of homogeneity in both data instances and features, thus also achieving clustering and dimensionality reduction simultaneously. Students and features could be modelled as a bipartite graph and a simultaneous clustering could be posed as a bipartite graph partitioning problem. In this paper we integrate an effective bagging strategy with Co-Clustering and present results for prediction of out-of-tutor performance of students. We report that such a strategy is very useful and intuitive, even improving upon performance achieved by previous work.

## Keywords

Out-of-Tutor Prediction, Dynamic Assessment, Spectral Co-clustering, Ensemble Learning, Bootstrap-Aggregation

## 1. INTRODUCTION

A significantly large student population would usually have a wide variation in learning rates and knowledge levels. While there are numerous reasons for this diversity, three major reasons are related to: the type of instruction or help they respond best to, the way they are oriented towards learning and their levels of intellectual development [1],[2]. Needless to say, such differences would be reflected in the way students interact with educational software, making educational data quite difficult to mine well. Specifically there are many educational data mining problems where the end goal is to predict the performance of a student on a given in-tutor or out-of-tutor task. In-tutor tasks include predicting the probability that a student will answer an item correctly after attempting a sequence of similar questions whereas out-of-tutor tasks include being to predict student performance in post-tests based on the data from their tutor usage.

The idea that students are quite different makes it apparent that perhaps it is not such a good idea to fit a global prediction model over the entire dataset for making predictions. In spite of the differences between students, educators commonly observe that students actually lie in very rough groups and have similar pedagogical needs. Taking a cue from this intuition, the task of prediction can be improved by clustering students into somewhat homogeneous groups and then training a separate predictor for each group. Such a predictor would obviously be a much better representative of students in that cluster as compared to a predictor which is fit on the entire dataset. For example, it makes sense to have a different model for students roughly classified as fast learners and a different model for slow learners than the same for both. This rather simple strategy of grouping students together and then modeling them separately can lead to improved performance in prediction and perhaps even better interpret-ability.

While the above approach is compelling, there are two major issues with it. Firstly, while it is useful to model students as belonging to different groups, it is also known that such groupings are quite fuzzy and approximate. Students might actually possess different characteristics in varying degrees and what really sets them apart are certain dominant characteristics. For example students classified as fast learners might actually be slow learners in certain skills. A fast learner might also belong to the group of students that are good at recalling information etc. Thus, such complex characteristics can not be possibly modelled by simply clustering

students to a certain limit and then training models for each cluster. This "spread" of features in a student across groups also needs to be captured to make a distributed predictive model such as the above more meaningful. Such an issue can be resolved by varying the granularity of the clustering and training separate models each time so that such features can be accounted for. A simple yet quite effective strategy to do so was proposed by the authors and was seen to work quite well both in educational contexts (in-tutor predictions [3], out-of-tutor predictions [4],[5]) and more generally [6].

The second problem with the above approach is that clustering is implicitly suggested to be one-way i.e only clustering students. But this need not necessarily be the case and only clustering students would consider only half of the story. As an example, consider a matrix in which the rows represent students and the columns represent their responses to certain items. Clearly, clustering students would depend upon their item distributions, implicitly suggesting that for certain students certain items are more important than others. Similarly if items were to be clustered, they would depend on which groups of students get them correct (or incorrect) most frequently. This indicates a duality between these two clusterings, which on simultaneous co-clustering could be very useful in answering many research questions. Co clustering of such a student versus item matrix would pair clusters of student proficiency to clusters of item performance which could be seen as a sort of a subject treatment interaction. This idea could be extended to the more general case of students and features rather than just items. In this work we use this idea of co-clustering students and their tutor interaction features and interleave it with the bagging strategy which was used with clustering [3],[4],[5],[6]. This combined approach is then used to predict the post-test scores of students.

This paper is organized as follows: In Section 2 we discuss the idea of co clustering in more detail and that co clustering could be posed as a bipartite graph partitioning problem. In Section 3 we describe a general framework in which we interleave co clustering with the idea of generating an ensemble. In Section 4 we describe the experimental results which demonstrate the validity of this approach. In Section 5 we discuss the results and also describe some avenues for further work.

# 2. CO-CLUSTERING

Clustering is a fundamental tool from unsupervised learning for data analysis that groups together relatively homogeneous objects. The central idea for clustering is that every object could be specified by a feature vector (or a point in the feature space) and then the degree of homogeneity between them could be measured by some objective function that uses these feature vectors. For example in k-means clustering: the points are grouped so as to minimize a distortion function, which is basically the sum of distances of all points from their assigned cluster centroids [7].

Clustering algorithms are one-way, i.e. one dimension of the data (say the rows of the data matrix) is clustered based on the similarities measured on the second dimension (say the columns). As pointed out in the previous section it might be desirable, quite frequently, to cluster along both the dimensions simultaneously, exploiting the apparent duality between them. Such simultaneous clustering can often offer interesting insights about the nature of interaction between the clusters at both the dimensions [8]. This utility is fast making co-clustering a fundamental tool for data analysis as is indicated by its widespread use in text and document mining [9], [10]; bioinformatics and gene expression analysis [11], [12]; collaborative filtering [13] and many others practical applications.

While there are now a number of approaches to co-clustering such as based on spectral graph theory [10] and information theory [14], [15], each with its advantages, we consider the approach proposed by Dhillon [10] which formulates the problem of co-clustering as a bipartite graph partitioning problem. We now briefly describe this approach starting with the relevant notation and definitions.

## 2.1 Notation and Definitions

A graph is represented as $G = (\mathcal{V}, E)$ where $\mathcal{V}$ represents the set of vertices and $E$ represents the set of all edge weights $E_{ij}$, where $E_{ij}$ is the edge weight between vertices $\{i, j\}$.

*Definition 1.* The $n \times n$ **Weighted Adjacency Matrix** of an undirected graph is defined as the matrix $(m_{ij})_{i,j=1,\ldots,n}$. If $m_{ij} = 0$ it implies that vertices $v_i$ and $v_j$ are not connected by an edge. If $m_{i,j} \neq 0$ it implies that the vertices $\{i, j\}$ are connected and $m_{i,j}$ is the corresponding edge weight. Since the graph is undirected, $m_{ij} = m_{ji}$ necessarily.

*Definition 2.* Given the weighted adjacency matrix of a graph and a partition of the vertex set $\mathcal{V}$ into two disjoint subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, the **cut** between these two subsets is defined as:

$$cut(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} M_{ij}$$

An undirected **bipartite graph** is a triple represented by $G = (\mathcal{S}, \mathcal{F}, E)$ where $\mathcal{S}$ and $\mathcal{F}$ are two sets of vertices and $E$ is the set of edges. Since it is a bipartite graph one end of the edges in set $E$ have an endpoint in $\mathcal{S}$ and another in $\mathcal{F}$. In our case the set $\mathcal{S}$ is the set of students while the set $\mathcal{F}$ is the set of features. The set of features could readily be seen as a set of item-responses as well. If $\mathcal{F}$ is the set of items, then an edge between $s_i$ and $f_j$ exists if that item was answered correctly by a student and not otherwise. More generally, if $\mathcal{F}$ is just a set of features, then the edge $\{s_i, f_i\}$ simply represents the value of that feature scaled between 0 and 1 for that student. Given this definition of a Bipartite Graph, now we define the adjacency matrix of the same.

Consider a $m \times n$ dimensional data matrix with students on the rows and the items or features on the columns. Let's suppose this matrix is given by $A$. Clearly, the adjacency of the bipartite graph is given as:

$$M = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

The zeroes on the top-left and the bottom-right sub-matrices signify the absence of connections amongst the elements of $\mathcal{S}$ and $\mathcal{F}$ respectively (since connections in a bipartite graph can only run between $\mathcal{S}$ and $\mathcal{F}$). The matrix $M$ is represented such that taking $A$ at the top right corner and $A^T$ at the bottom left implies that the first $m$ rows of $M$ represent the set of students and the next $n$ rows represent the set of features or items.

Suppose the Bipartite Graphs (whose adjacency matrix is defined above) is partitioned into $k$ clusters $\mathcal{V}_1, \ldots, V_k$. Given this partitioning, a corresponding set of student clusters $\mathcal{S}_1 \ldots \mathcal{S}_k$ and corresponding feature clusters $\mathcal{F}_1 \ldots \mathcal{F}_k$ would also be obtained. It could be intuitively seen that the best possible such set of clustering for all such pairs would be when the sum of all edges which cross between clusters is the minimum possible. As defined by [10] this corresponds to:

$$cut(\mathcal{S}_1 \cup \mathcal{F}_1, \ldots, \mathcal{S}_k \cup \mathcal{F}_k) = min_{\mathcal{V}_1,\ldots,\mathcal{V}_k} cut(\mathcal{V}_1, \ldots, V_k)$$

Where $\mathcal{V}_1, \ldots, \mathcal{V}_k$ represents a k-partitioning of the graph. The above definition leads us to the Bipartite Graph Partitioning problem:

*Definition 3.* **The bipartite graph partitioning problem:** Given a graph as defined earlier and subsets of $\mathcal{V}$ which are almost of equal size, say $\mathcal{V}_1^*$ and $\mathcal{V}_2^*$. The required partition is

$$cut(\mathcal{V}_1^*, \mathcal{V}_2^*) = min_{\mathcal{V}_1,\mathcal{V}_2} cut(\mathcal{V}_1, \mathcal{V}_2)$$

The bipartite graph partitioning problem as defined above is NP-Complete. However, a good relaxation to this problem is given by spectral graph bi-partitioning. This relaxation is achieved via the graph Laplacian. The laplacian $L$ of a graph is a symmetric positive semi-definite matrix such that its un-normalized form is given by $L = D - M$ where $D$ is the degree matrix and $M$ is the adjacency matrix as defined earlier. Note that $D$ is only a diagonal matrix while $M$ is a symmetric matrix with all zeros in the diagonal. Thus, the Laplacian encodes both $D$ and $M$ in it and has many useful properties such as being positive semi-definite, which make it very useful for tasks such as clustering [24]. One property of the Graph Laplacian that make it particularly suitable for clustering are related to the properties of its spectrum. The spectra of the Graph Laplacian unfolds the data manifold to give an lower dimensional embedding which can give "better" clustering results.
Returning to the Bipartite Graph Partitioning Problem, as demonstrated by Dhillon [10] and Mohar [24], the second eigenvector of the generalized eigenvalue problem $Lz = \lambda Dz$ gives a real relaxation to the problem of finding the minimum normalized cut $\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2)$. The normalized cut is basically a cut that favours finding balanced partitions i.e. if the cut of two different partitions is the same, then the normalized cut is smaller for that partition which is more balanced. Thus it favours partitions that are balanced and have a small cut value. Clearly, the normalized cut is more suitable for tasks such as clustering [16]. Note that this relates to the ideas above relating to the optimal bi-partitionings in the following way: We want balanced clusterings with minimum cut for solving the bipartite graph partitioning problem, which would also be the optimal clustering for us. Thus looking at the Laplacian of the bipartite graph might provide such a clustering.

## 2.2 Spectral Co-Clustering

Given the definitions and notions in the previous section, in this section we state an algorithm [10] for finding the optimal co-clusters $\{\mathcal{S}_1 \cup \mathcal{F}_1\}, \ldots, \{\mathcal{S}_k \cup \mathcal{F}_k\}$ as mentioned above. For that we define the graph laplacian of a bipartite graph as such an optimal clustering can be found using a laplacian. Using the definition of $L = D - M$ as defined above and also the definitions of $D$ and $M$. The laplacian may be written as:

$$L = \begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix}$$

and

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$$

where $D_1$ and $D_2$ correspond to the degree matrices of $A$ and $A^T$ respectively.
If the generalized eigenvalue problem $Lz = \lambda Dz$ is written for the above laplacian for a bipartite graph and then rearranged, it has been demonstrated [10] that the resulting equations define the equations for a singular value decomposition of the normalized matrix

$$A_n = D_1^{-1/2} A D_2^{-1/2}$$

Thus instead of finding the second smallest eigenvector corresponding to the second eigenvalue, one could find the left and the right singular values in its place. Finding the right singular value gives a bi-partitioning of students while the left singular value gives a bi-partitioning of the features. These can then be used to find the optimal bi-partition as defined above.

**Algorithm 1.**

1. *Given the co-occurrence or data matrix scaled to between* 0 *and* 1 *A, form the normalized matrix.*

$$A_n = D_1^{-1/2} A D_2^{-1/2}$$

2. *Compute the second left and right singular vectors for* $A_n$*, concatenate them together to form a vector* $z$*.*

3. *Run k-means on this vector to obtain a simultaneous clustering of both the students and the features.*

This algorithm can be extended to a multipartition case if instead of finding the second singular values, the first $log_2(k)$ singular vectors are found. The rest of the process remains the same.
Note that this algorithm gives a simultaneous clustering of the rows and the columns and is restricted in the sense that the number of row and columns clusters have to be the same. We modify this by running k-means two times. If the number of row clusters is $k$ and then the number of column vectors is $l$, then we run k-means on the vector $z$ twice, once to find $k$ clusters and then to find $l$ clusters. The first $m$ elements of the length $m + n$ cluster assignment vector run will then correspond to the row clusters and the last $n$ elements of the cluster assignment vector in the second run will correspond to the column cluster indices.

# 3. BAGGING STRATEGY

The statement of the supervised learning problem in machine learning could be roughly stated as follows: Given a training set consisting of ordered pairs of feature vectors and

their associated labels (which might be discrete or continuous), the task of a learning algorithm is to learn a functional map from the feature space to label space. A learning algorithm is said to be more powerful if it is able to learn mappings such that it can generalize well and make correct predictions on test data-points on which it was not trained. Since the functional map under consideration might be highly non-linear, learning algorithms that output only a single mapping (frequently referred to as the hypothesis) might suffer from statistical, computational and representation issues that restrict them from learning good mappings. One way of solving this problem is to transform the feature space into a more suitable and "richer" representation such that learning using this new representation gives much better functional maps as compared to the original representation. This is the motivation behind deep learning methods which have caused a new wave of excitement in the machine community since 2006 [17]. Another way of solving this problem atleast partly, is by using ensemble learning methods [18],[19],[20]. The basic idea behind ensemble methods is that they involve running a "base learning algorithm" multiple times, each time with some change in the representation of the input (e.g. only considering a subset of features in each run) so that a number of diverse predictions (or maps) could be obtained. This diversity in prediction is then exploited to get better predictions. Thus ensemble methods approach the said problem by both trying to learn multiple functional maps and also by learning a more distributed and hence "richer" representation of the input space at the same time. In the next section we describe a method to use clustering for bootstrapping.

## 3.1 Clustering for Bootstrapping

In earlier work we introduced the idea of using clustering for bootstrapping [3], [4], [5], [6]. This idea was quite unlike other bagging methods which use a random subset to bootstrap. Thus, it had the potential advantage that the subsets used to bootstrap could be more interpretable. Before we generalize this methodology using co-clustering we first briefly describe the methodology using clustering.

The training set was first clustered into $k$ disjoint clusters. A linear regression model was trained on each of the clusters only based on the training points assigned to that cluster. Since each such linear regression was a representative of only one cluster, we called it a cluster model. Thus, for a given $k$, there would be $k$ cluster models. But since all the clusters are mutually exclusive, the training set is represented by all the cluster models taken together. This is called a prediction model ($PM_k$). For an incoming test point on which a prediction is to be made, we first identify the cluster that point belongs to. After the cluster has been identified, the appropriate cluster model could be used to make a prediction for that point. Now note that we don't specify the number of clusters in the above. Hence, we can change the granularity of the clustering from 1 to some high value, say $K$. In each instance we would get a different prediction model (a special case would be $PM_1$, which would basically be when one linear regression model is trained on the entire dataset). Thus, we would obtain a set of $K$ prediction models each of which would make a separate prediction on the test set. Since we vary the granularity of the clustering, each of these predictions are different, this diversity in prediction could be used by averaging all the (or half) the predictions obtained to get a single much stronger prediction.

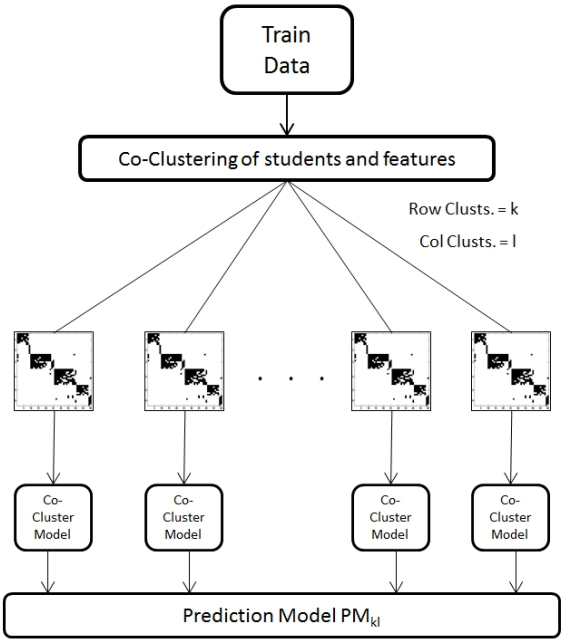



Figure 1: Finding a Prediction Model, $PM_{kl}$ with $k$ row clusters and $l$ column clusters

## 3.2 Co-Clustering for Bootstrapping

Note that the clustering is only one-way. That is, bootstrapping is done by only changing the data instances available for each cluster model (by changing the number of cluster models itself) but the number of features used in each case is the same. A cluster basically is a bunch of rows in the data matrix with all columns. A co-cluster on the other hand would be a "block" in the data matrix with a sub-set of rows and a sub-set of columns assigned to each "co-cluster". Thus a co-clustering could be thought of as a simultaneous clustering and dimensionality reduction of the data. Note that a clustering is only a special case of co-clustering when the columns are not clustered at all (or have only one column cluster).

Clearly, the above bagging methodology can be suitably modified using co-clustering. For a given number of row clusters $k$ and column clusters $l$ we could have $k$ co-clusters where-in each cluster has only some features assigned to it (note that the definition is symmetric i.e we could think of this as $l$ co-clusters). For each co-cluster we train a separate linear regression model only using the data instances and features assigned to it. We thus obtain $k$ Co-Cluster Models. Like in the above case for clustering, the combination of the $k$ co-cluster models would be considered to be a Prediction Model which makes a single prediction on the test set. We can then vary $k$ from 1 to some value $K$ and $l$ from 1 to some value $L$. By doing so, we would get a total of $K \times L$ prediction models. We then average a subset of the predictions made by these models to obtain a much stronger prediction.

There are some interesting aspects to such a methodology using co-clustering. For $k = 4$ and $l = 4$, the grid in Figure

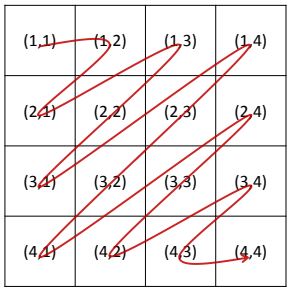



Figure 2: Ordering the Co-Cluster Prediction Models, $PM_{kl}$

2 illustrates all the Prediction Models ($PM_{kl}$) that could be obainted by co-clustering. The Prediction Model $PM_{1,1}$ represented by $(1, 1)$ is simply the case when there is one data cluster and only one feature cluster i.e the original data matrix itself. The prediction model for this case would simply be training a linear regression on the entire dataset, considering all the features. The first column of this grid represents the case when the number of feature clusters is just one, while the number of row clusters are changed. Note that this is simply the methodology described above in Section 3.1 using clustering. The first row of this grid is also equally interesting. In this case the number of row clusters is always one i.e the entire dataset is considered in all co-clusters, while the column clusters are successively changed. It should be noted that this is a sort of a step-wise regression, where a linear regression is trained on the entire dataset but the number of features that are used to train it are changed (usually reduced as $l$ increases). All the other cases are a cross between these two extreme cases. We see that it seems plausible that a bagging strategy using co-clustering if averaged properly could definitely have more predictive power as it generates diversity by considering a different subset of data instances and features each time, consequently also generating a much larger set of predictions.

### 3.3 Blending Predictions

As mentioned before, the method for combining the predictions returned by the various prediction models is a naive averaging strategy. When the prediction models were generated by clustering ($PM_k$), we either averaged the first $K/2$ predictions (where $K$ was the maximum number of clusters) [6] or we learned the best number of prediction models that could be averaged by an internal cross-validation [6]. The averaging idea is not immediately straightforward when co-clustering is used to generate the prediction models. This is because the prediction models are obtained by changing two parameters. It is also observed that prediction models with a high $k$ or $l$ return poor accuracies, thus it wouldn't be useful to average predictions from all the $PM_{k1}$ models first and then $PM_{k2}$ models and so on (i.e. traversing the grid row-wise or column-wise). Since high values of $k$ and $l$ are counter-productive, we take the order of the prediction models such that the sizes of $k$ and $l$ increase uniformly. This ordering is illustrated by the curve in Figure 2. The first half of this reordered set of predictions are then averaged.

## 4. EXPERIMENTAL VALIDATION

In this section we report experimental results for using co-clustering for bagging and compare results with the benchmark ($PM_{11}$) and clustering alone.

### 4.1 Dataset Description and Context

We primarily experiment with two datasets in this study. This data was collected to study if dynamic assessment, which has long been advocated as an effective method for assessment, was actually better than the traditional static assessment [21], [22]. Dynamic assessment is an interactive approach to student assessment which is primarily based on how much help a student requires during a practice test. Traditional static testing only takes into account the percentage of questions that the student gets correct. Feng *et al.* [23] showed that features that only recorded how much assistance a student got while interacting with a tutor alone were better predictors of student performance in post-tests held later in the year as compared to how many questions students got correct. This was confirmed in subsequent studies [4], [5]. Thus if Co-Clustering is able to improve predictions, then this study could further lend weight to the idea that dynamic testing is indeed better than static testing and that we could further improve upon $PM_{11}$. It must be noted that $PM_{11}$ would correspond to results reported in [23] which were better than static assessment. $PM_{11}$ basically corresponds to the condition when all the dynamic features are considered and all of the training set is used to train a predictor.

The datasets come from the 2004-05 and 2005-06 school years, the first two full years when ASSISTments.org was used in schools in Massachusetts. ASSISTments is an e-learning tutoring system developed at Worcester Polytechnic Institute which assesses students as it assists. These datasets contain features that measure the interaction of students with the tutor and their actual final grades, which they obtained at the end of the year in the Massachusetts state test (MCAS). There a total number of six features in these datasets **1) DA Original Count** is the number of questions that the students answered with assistance in the dynamic condition. **2) DA Original Percent Correct** is the percent of questions of feature 1 that students get correct . **3) DA Scaffold Percent Correct** is the percentage on tutorial help questions that students get correct. **4) DA Average Time** is the average time that a student spends on a question **5) DA Average Attempt** is the average number of attempts students made per question. **6) DA Average Hints** is the average number of hints that students used. The task is to use these interaction features to predict the MCAS scores that students might get at the end of the school year. The static condition feature is percentage of questions answered correct in static testing. This feature is never used for making predictions for the dynamic condition. The data in the 2004-05 set (ASSISTments 2004-05) is for 628 students, while the 2005-06 data (ASSISTments 2005-06) is for 761 students.

For experimentation we do a five fold cross-validation on the dataset and report results for the base condition ($PM_{11}$) and the various blended results which were obtained by averaging as discussed in Section 3.3. For the sake of comparison we also include results with k-means clustering too. In both cases we consider the ensembled results, with the top $K$ predictions averaged as described in [4], [5] and also in Section 3.1. Following results in [4] and [5] we report results in terms of the mean absolute difference (MAD).

Finally, for pre-processing: As mentioned in Section 2, to

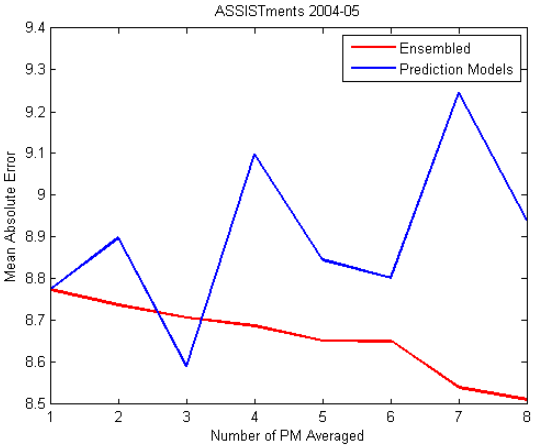



Figure 3: Performance on the 2004-05 Set

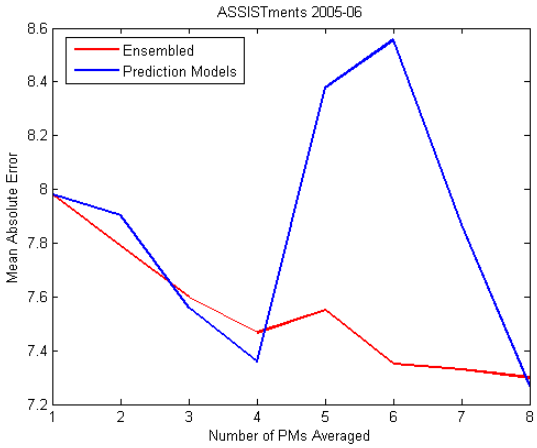



Figure 4: Performance on the 2005-06 Set

obtain a bipartite partitioning $A$ must contain values that are either binary or scaled between 0 and 1. Thus, in each fold each feature column is scaled to between 0 and 1 so that $A_n$ could be considered a co-occurrence matrix. This marks a slight difference from earlier papers in which the feature scaling was done so as to map all the data-points to between $-1$ and 1 by using the mapminmax command of MATLAB. This slight difference might result in a small variation in the results.

## 4.2 Experimental Results

We first report results on the ASSISTments 2004-05 dataset. The five fold cross-validated results using co-clustering are reported in Figure 3. The number of row clusters ($k$) and the number of column clusters ($l$) were restricted to 4 each. This resulted in 16 prediction models. The x-axis in the graph represents the first eight prediction models on doing co-clustering, while the y-axis simply gives the mean absolute error. We observe that the accuracy of co-clustering alone is quite bad (as seen by the blue line) as compared to the baseline ($PM_{kl}$, which is basically the result for x = 1 in this graph. Note that the baseline is the dynamic condition of Feng [23]). These predictions are those given by the first elements of the ordered set of co-cluster prediction models as defined in Section 3.3. However, averaging these prediction models successively gives better and better predictions (as can be seen by the red line).

Similar results were reported in the ASSISTments 2005-06 dataset as shown in Figure 4. In this dataset the prediction models are far worse than the ensembled results as compared to the previous dataset. Again, we obtain 16 prediction models after co-clustering and successively average the first eight (the first with second, the first with second and third and so on) after they have been arranged in the way suggested in Section 3.3. Again the ensembled results do much better over the baseline (we report exact figures and significance in Tables 1 and 2).

In Table 1 we compare the mean absolute errors when predictions of the first five prediction models are bagged. We report results when the Prediction Models are obtained both by using co-clustering and using k-means clustering on the ASSISTments 2004-05 dataset. The figures in bold indicate statistical significance over the baseline prediction on a paired t-test. Results in Table 2 compare the predictions obtained by using co-clustering and k-means for bagging on the ASSISTments 2005-06 dataset.

Table 1: Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2004-05 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline

| Pred. Models | Co-Clust | k-means |
|---|---|---|
| 1 | 8.7741 | 8.7741 |
| 2 | 8.7379 | 8.7518 |
| 3 | **8.7087** | **8.6725** |
| 4 | **8.6879** | 8.7153 |
| 5 | **8.6574** | **8.7100** |

The results are significantly better over the baseline and also indicate that the dynamic assessment condition returns a much better prediction of student test scores as compared to the static condition. It has already been noted that the static test condition results are significantly worse as compared to even the baseline by [23] and [4], and thus we don't report results for the static condition.

# 5. DISCUSSION AND FUTURE WORK

The datasets that were used for the validation of this bagging technique, which is based on co-clustering were not very large and did not have a large number of columns. Thus,

Table 2: Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2005-06 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline

| Pred. Models | Co-Clust | k-means |
|---|---|---|
| 1 | 7.9822 | 7.9822 |
| 2 | **7.7716** | **7.8185** |
| 3 | **7.5990** | **7.8034** |
| 4 | **7.4680** | **7.7815** |
| 5 | **7.5503** | **7.6487** |

these results were initially surprising. One would imagine that in a dataset which has a small number of features, perhaps a feature selection might not be too helpful. However, our experiments show us otherwise. The results that we obtain, while modest improvements show that this technique though simple can give access to a novel source of variance in the data. It can potentially also have some nice properties in terms of returning simpler and more interpretable groups. For example, it was earlier pointed out that one row of the prediction models were actually nearly like a linear regression model in which the features are successively eliminated. At the same time it was observed that one column of the prediction models were actually just the various prediction models that we obtained on clustering alone as reported in some previous work. It would be interesting to see how the Co-Clusters (which are basically blocks in the data matrix) on a student-item dataset would pair clusters of student proficiency to clusters of item performance which could be seen as a sort of a subject treatment interaction.

In the literature, it has been said that the real strength of co-clustering is with binary valued data, co-occurrence tables and basically in scenarios which involve collaborative filtering. Hence, datasets which are basically a student by item matrix would be an ideal candidate for trying out this technique. In the KDD Cup 2010 Töscher and Jahrer modelled student response data as a collaborative filtering task and used matrix factorization techniques for the same. Given the connections of co-clustering with matrix factorization, it is worth investigating how useful it could be in such a setting.

In [3], the authors clustered students based on tutor interaction features and then trained separate Knowledge Tracing models for students based on the cluster they were in. This was done so because it was not possible to cluster the item sequences directly and an indirect approach had to be taken. This co-clustering technique seems to give an alternative by which such matrices might be clustered more readily without the need to cluster the tutor interaction features.

In summary, in this paper we propose a bagging technique that uses co-clustering and demonstrate that it's performance is better than that obtained by bagging using clustering. We also suggest that it is most suitable for datasets which are like co-occurrence tables and believe that it would be a good direction for future work since such student-item datasets are usually of this form.

## 6. ACKNOWLEDGEMENTS

The authors are indebted to all the funders listed over here `http://www.webcitation.org/5xp605MwY`. The research of the third author (Gábor Sárközy) is supported in part by the National Science Foundation under Grant No. DMS-0968699.

## 7. REFERENCES

[1] Felder, R.M., and Brent. R.,Understanding Student Differences, Journal of Engineering Education, Vol. 94, No. 1, 2005, pp. 57-72.

[2] Bransford, J.D., Brown, A.L., and Cocking, R., eds., How People Learn: Brain, Mind, Experience, and School, Washington, D.C.: National Academy Press, 2000.

[3] Pardos, Z. A., Trivedi, S., Heffernan. N. T., and Sárközy. G. N., Clustered Knowledge Tracing, In the Proceedings of the 11th International Conference on Intelligent Tutoring Systems 2012, Chania, Greece.

[4] Trivedi, S., Pardos, Z. A., Heffernan, N. T., Clustering Students to Generate an Ensemble to Improve Standard Test Score Predictions, G. Biswas et al. (Eds.): AIED 2011, LNAI 6738, In The proceedings of the 15th International Conference on Artificial Intelligence in Education 2011, Auckland, New Zealand, pp. 377-384.

[5] Trivedi, S., Pardos, Z. A., Sárközy, G. N., Heffernan, N. T., Spectral Clustering in Educational Data Mining, In Proceedings of the 4th International Conference on Educational Data Mining 2011, Eindhoven Netherlands, pp. 129-138.

[6] Trivedi, S., Pardos, Z. A., and Heffernan, N. T., The Utility of Clustering in Prediction Tasks, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics (under review).

[7] Hartigan, J. A., Wong, M. A., Algorithm AS 136: A K-Means Clustering Algorithm, Journal of the Royal Statistical Society, Series C (Applied Statistics), 1979, 28 (1): pp. 100-108.

[8] Hartigan, J. A., Direct Clustering of a Data Matrix, Journal of the American Statistical Association, 67(337): pp. 123-129, 1972.

[9] Dhillon, I. S., S, Mallela., and Kumar, R., A divisive Information-Theoretic Feature Clustering Algorithm for text classication. Journal of Machine Learning Research, 3(4): pp. 1265-1287, 2003.

[10] Dhillon, I. S., Co-clustering documents and words using bipartite spectral graph partitioning. In Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining, pp. 269-274, 2001.

[11] Hanisch, D., Zein, A., Zimmer, R., Lengauer, T., Co-clustering of biological networks and gene expression data. Bioinformatics 18 (Suppl.), 2002, pp. 145-154.

[12] D'Haeseleer, P., Liang, S., and Somoyogi, R., Genetic Network Inference: From co-expression clustering to reverse engineering, Bioinformatics, 16, 2000, pp. 707-726.

[13] George, T., and Merugu, S., A scalable collaborative ltering framework based on co-clustering. In Proceedings of the IEEE Conference on Data Mining, pp. 625-628, 2005.

[14] Dhillon, I., Mallela, S., and Modha, D., Information-theoretic co-clustering. In Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD), pp. 89-98, 2003.

[15] Banerjee, A., Dhillon, I. S., Ghosh, J., Merugu, S., and Modha, D. S, A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation, Journal of Machine Learning Research, 8, pp. 1919-1198, 2007.

[16] J. Shi, and J. Malik, Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (8), pp. 888-905, 2000.

[17] Bengio, Y., Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1): pp. 1-127, Now Publishers, 2009.

[18] Dietterich, T.G., Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) First International workshop on Multiple Classifier Systems. LNCS, pp. 1-15. Springer, New York, 2000.

[19] Dietterich, T.G., An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, Machine Learning 40, pp. 139-157, 2000.

[20] Breiman, L., Random Forests, Machine Learning 45(1), pp. 5-32, 2001.

[21] Grigerenko, E.L., Steinberg, R.J.: Dynamic Testing. Psychological Bulletin 124, pp. 75-111, 1998.

[22] Campione, J. C., Brown, A. L.: Dynamic Assessment: One Approach and some Initial Data. Technical Report. No. 361. Cambridge, MA. Illinois University, Urbana, Center for the Study of Reading. ED 269735, 1985.

[23] Feng, M., Heffernan, N.T., Koedinger, K.R.: Addressing the assessment challenge in an online system that tutors as it assesses. User Modeling and User-Adapted Interaction: The Journal of Personalization Research 19(3), 2009.

[24] Mohar, B., The Laplacian spectrum of graphs. In Graph theory, combinatorics, and applications. Vol. 2 (Kalamazoo, MI, 1988), New York: Wiley, 1991, pp. 871-898.

# Comparison of methods to trace multiple subskills: Is LR-DBN best?

Yanbo Xu
Carnegie Mellon University
RI-NSH 4105
5000 Forbes Ave, Pittsburgh, PA 15213
yanbox@cs.cmu.edu

Jack Mostow
Carnegie Mellon University
RI-NSH 4103
5000 Forbes Ave, Pittsburgh, PA 15213
mostow@cs.cmu.edu

## ABSTRACT
A long-standing challenge for knowledge tracing is how to update estimates of multiple subskills that underlie a single observable step. We characterize approaches to this problem by how they model knowledge tracing, fit its parameters, predict performance, and update subskill estimates. Previous methods allocated blame or credit among subskills in various ways based on strong assumptions about their relation to observed performance. LR-DBN relaxes these assumptions by using logistic regression in a Dynamic Bayes Net. LR-DBN significantly outperforms previous methods on data sets from reading and algebra tutors in terms of predictive accuracy on unseen data, cutting the error rate by half. An ablation experiment shows that using logistic regression to predict performance helps, but that using it to jointly estimate subskills explains most of this dramatic improvement. An implementation of LR-DBN is now publicly available in the BNT-SM student modeling toolkit.

## Keywords
Conjunctive knowledge tracing, Dynamic Bayes Nets, logistic regression

## 1. INTRODUCTION
Knowledge tracing (KT) [1] is widely used to update an intelligent tutor's estimate of the probability that a student has a given skill, based on the student's observable performance on steps that use the skill. KT does not in itself address the issue of how to update multiple subskills used in the same step. This paper compares various approaches to this "multiple subskills problem." Section 2 frames the space of prior (and new) methods. Section 3 describes a recent method named LR-DBN [2]. Section 4 compares LR-DBN against previous methods on two data sets. Section 5 concludes.

## 2. COMPARATIVE FRAMEWORK
As a framework to compare previous and proposed methods for tracing multiple subskills, we use four aspects to characterize them: how they **represent** the KT model, how they **fit** the model parameters to observations of multi-subskill steps, how they use the model to **predict** performance on such steps, and how they **update** estimates of the subskills based on observed performance.

### 2.1 Represent model
Previous solutions represent the student's knowledge at step *n* as a hidden state $K^{(n)}$ in a Hidden Markov Model (HMM), shown in Figure 1. It has knowledge parameters *already know* for the probability that the student knew the skill to start with, *learn* for the probability of the transition from not knowing the skill to knowing it, and *forget* (usually assumed to be 0) for the probability of the transition from knowing the skill to not knowing it. It also has performance parameters *guess* for the probability $P(C^{(n)} \mid not\ K^{(n)})$ of performing the step correctly despite lacking the skill, and *slip* for the probability $P(not\ C^{(n)} \mid K^{(n)})$ of performing the step incorrectly despite knowing the skill.

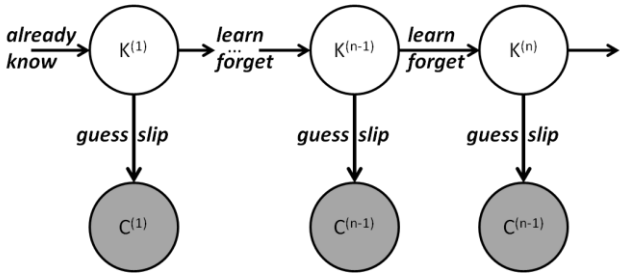



**Figure 1: Single-skill knowledge tracing architecture**

### 2.2 Fit parameters
One previous solution [3] tries to sidestep the problem by modeling each set of subskills as a distinct individual skill, e.g., computing the area of a circle embedded in a figure vs. by itself. However, modeling different sets of subskills as independent skills ignores transfer of learning between them.

Other previous solutions [4-6] simply treat each subskill used in a step as if it were entirely responsible for that step. They train a separate KT model for each subskill on observations of all the steps that use it. Thus the same observed step appears in the training data for every subskill that it uses. They simply estimate the model parameters for each subskill using the same training procedure as for conventional KT.

### 2.3 Predict performance
In standard "single-skill" KT, predicting performance $C^{(n)}$ at step *n* involving skill *j* is simple:

$$P\left(C^{(n)}\right) = P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j$$

**Equation 1: Standard KT prediction**

Previous solutions to the multiple-subskills problem predict performance on a step by combining in different ways the probabilities of correctly performing all the skills it requires. One way, based on an assumption that they are probabilistically independent, multiplies them [4, 6]:

$$P\left(C^{(n)}\right) = \prod_j P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j$$

**Equation 2: Independent subskills performance prediction**

The weakest-subskill alternative [5] takes their minimum:

$$P\left(C^{(n)}\right) = Min_j\, P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j$$

**Equation 3: Weakest-subskill performance prediction**

## 2.4 Update estimate

To update its estimate of a skill *j* based on the observed success of a step *n* that uses it, standard KT applies Bayes' rule:

$$P_{posterior}\left(K_j^{(n)}\right) = P\left(K_j^{(n)} \middle|\, C^{(n)}\right) = \frac{P\left(K_j^{(n)}\right) \times (1 - slip_j)}{P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j}$$

**Equation 4: Standard KT skill update for successful step**

Conversely, the standard update rule if the step fails is:

$$P_{posterior}\left(K_j^{(n)}\right) = P\left(K_j^{(n)} \middle|\, not\ C^{(n)}\right) = \frac{P\left(K_j^{(n)}\right) \times slip_j}{P\left(K_j^{(n)}\right) \times slip_j + \left(1 - P\left(K_j^{(n)}\right)\right) \times (1 - guess_j)}$$

**Equation 5: Standard KT skill update for failed step**

Either way, it estimates the probability of knowing the skill at the next step as either knowing without forgetting, or learning:

$$P\left(K_j^{(n+1)}\right) = P_{posterior}\left(K_j^{(n)}\right) \times (1 - forget_j) + \left(1 - P_{posterior}\left(K_j^{(n)}\right)\right) \times learn_j$$

**Equation 6: Standard KT next-step update**

When a step involves multiple subskills, previous methods use different ways to allocate responsibility among them for the observed success or failure of the step. The "full responsibility" approach applies these equations to all the subskills. The "update weakest subskill" approach simply applies the standard update equations above to whichever subskill in a step has the lowest probability, and leaves the others unchanged. Its "blame weakest, credit rest" variant credits the other subskills as correct even if the step failed.

Conjunctive knowledge tracing (CKT) [6] also predicts the probability of a step succeeding as a product of its subskill probabilities using Equation 2, and gives all of them full credit for success using Equation 4. However, rather than place full blame on each subskill for failure, CKT apportions blame among them differently. Instead of using Equation 5 to update each subskill based just on its own *guess* and *slip* probabilities, CKT takes into account those of the other subskills as well, as follows.

Bayes' rule says how to update a skill based on performance:

$$P_{posterior}\left(K_j^{(n)} \mid not\ C^{(n)}\right) = \frac{P\left(not\ C^{(n)} \mid K_j^{(n)}\right) \times\ P(K_j^{(n)})}{P(not\ C^{(n)})}$$

**Equation 7: Bayes' rule for skill update**

Conditioning on having skill *j* at step *n* reduces $P(K_j^{(n)})$ to 1, simplifying the numerator of Equation 9 to:

$$P\left(not\ C^{(n)} \mid K_j^{(n)}\right) = slip_j + (1 - slip_j) \times \prod_{i \neq j} \left[P(K_i^{(n)}) \times (1 - slip_i) + \left(1 - P(K_i^{(n)})\right) \times guess_i\right]$$

**Equation 8: CKT subskills update for failed step**

CKT computes the denominator by assuming independence:

$$P\left(not\ C^{(n)}\right) = 1 - P\left(C^{(n)}\right) = 1 - \prod_j \left[P(K_j^{(n)}) \times (1 - slip_j) + \left(1 - P(K_j^{(n)})\right) \times guess_j\right]$$

**Equation 9: CKT prediction based on multiple subskills**

Next, we introduce a different strategy of using logistic regression in KT to trace multiple subskills.

# 3. USING LOGISTIC REGRESSION TO TRACE MULTIPLE SUBSKILLS

We now describe two newer methods that trace multiple subskills using logistic regression. Previous KT methods fit their parameters independently using the same algorithm as for single-skill KT, thereby implicitly assigning full and equal responsibility to all the subskills in an observed step, and predict performance based on the weakest subskill or by multipying subskill estimates. Section 3.1 describes LR-DBN, which changes both these aspects by using Expectation Maximization (EM) [7] to fit parameters for multiple subskills simultaneously, and by using logistic regression to predict performance. As an ablation experiment to shed light on the relative impact of these two innovations, Section 3.2 introduces LR-DBN Minus, a hybrid method that fits the standard KT model just as previous methods do, but uses logistic regression to do prediction.

## 3.1 LR-DBN

LR-DBN is a recent but published method [2, 8] to trace multiple subskills, so we summarize it here only briefly in terms of the four aspects discussed in Section 2.

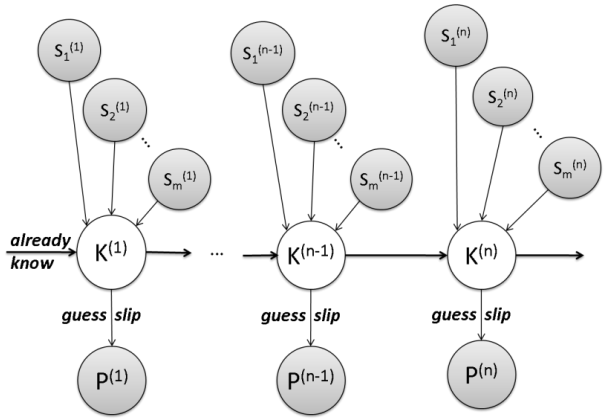



**Figure 2: Knowledge tracing with logistic regression**

**Represent model:** Like standard KT, LR-DBN represents the knowledge for step *n* as a hidden knowledge state $K^{(n)}$ in a dynamic Bayes net. However, as Figure 2 illustrates, LR-DBN

adds a layer of observable states $S_j^{(n)}$ as indicator variables to represent whether step *n* involves subskill *j*: 1 if so, 0 if not.

LR-DBN uses logistic regression to model the initial hidden knowledge state at step *0* and the transition probabilities from step *n-1* to *n* as follows:

$$P(not\ K^{(0)}) = \frac{exp\ (-\sum_{j=1}^{m} \beta_j^{(0)}\ S_j^{(0)})}{1 + exp\ (-\sum_{j=1}^{m} \beta_j^{(0)} S_j^{(0)})}$$

$$P(not\ K^{(n)} \mid not\ K^{(n-1)}) = \frac{exp\ (-\sum_{j=1}^{m} \beta_j\ S_j^{(n)})}{1 + exp\ (-\sum_{j=1}^{m} \beta_j\ S_j^{(n)})}$$

$$P(not\ K^{(n)} \mid K^{(n-1)}) = \frac{exp\ (-\sum_{j=1}^{m} \gamma_j\ S_j^{(n)})}{1 + exp\ (-\sum_{j=1}^{m} \gamma_j\ S_j^{(n)})}$$

**Equation 10: Logistic regression to combine subskills**

These three conditional probabilities for each subskill replace the KT knowledge parameters *already know*, *learn*, and *forget***,** but LR-DBN retains KT's *guess* and *slip* parameters at each step.

**Fit parameters:** LR-DBN uses Expectation Maximization (EM) [7] to fit parameters for all subskills together.

**Predict performance:** LR-DBN uses logistic regression in a Dynamic Bayes Net to combine multiple subskills more flexibly than using Equation 2 to multiply their probabilities or Equation 3 to take their minimum, but it uses Equation 1 to predict expected performance based on estimated knowledge, *guess*, and *slip*.

**Update estimates:** LR-DBN uses the same Bayes rule as single-skill KT to update its estimate of the hidden knowledge state in Equation 4, Equation 5, and Equation 6.

## 3.2 LR-DBN Minus

LR-DBN Minus is a hybrid of LR-DBN and standard KT. It combines KT's single-skill fitting process with LR-DBN's update and prediction based on logistic regression. The key is to convert the probability of knowing a subskill into a coefficient in logistic regression. LR-DBN uses logistic regression to model the transition probabilities between knowledge states, as well as the relation of the knowledge state at each step to the subskills it involves [2]. Thus, given the set of subskills $\{S_j^{(n)}\}$ used at step n, a set of coefficients $\{\beta_j^{(n)}\}$ exists such that

$$P(K^{(n)}) = 1 - sigmoid\left(\sum_{j=1}^{m} \beta_j^{(n)} S_j^{(n)}\right)$$

**Equation 11: Logistic regression for the knowledge state**

If we assume step *n* requires only a single subskill *i*, then $S_j^{(n)} = 0$ for all the j's such that $j \neq i$, and transformations between the probabilities $P\left(K_i^{(n)}\right)$ and the coefficients $\beta_i^{(n)}$ are:

$$P\left(K_i^{(n)}\right) = 1 - sigmoid\left(\beta_i^{(n)}\right)$$

$$\beta_i^{(n)} = logit(1 - P\left(K_i^{(n)}\right))$$

**Equation 12: Transformation between probabilities and logistic regression coefficients**

To update the estimates, we need to distribute the update at each step that is calculated either from Equation 4 or Equation 5 to the subskill coefficients. We assume that the coefficient for each subskill changes by the same amount $\Delta\beta$ when updated:

$$P_{posterior}\left(K^{(n)}\right) = 1 - sigmoid(\sum_{j=1}^{m} (\beta_j^{(n)} + \Delta\beta)\ S_j^{(n)})$$

**Equation 13: Update coefficients in LR-DBN Minus**

Then the update of each subskill becomes:

$$P_{posterior}\left(K_j^{(n)}\right) = 1 - sigmoid((\beta_j^{(n)} + \Delta\beta)\ S_j^{(n)})$$

**Equation 14: Update subskills in LR-DBN Minus**

Next we still use the standard KT Equation 6 to update subskills at step *n+1*. Now we have successfully transformed LR-DBN to upate on standard KT parameters. Note that we replace the separate $guess_j$ and $slip_j$ parameters for each subskill *j* with uniform *guess* and *slip* for all the steps. The reason is that LR-DBN combines subskills to estimate the probability of the student knowing a step and then uses *guess* and *slip* to predict performance. In contrast, previous methods apply $guess_j$ and $slip_j$ to each subskill *j* before combining them to predict performance on the step.

# 4. EXPERIMENTAL EVALUATION

To compare LR-DBN and LR-DBN Minus to previous methods for tracing multiple subskills, we fit seven models to real data, summarized in Table 1: LR-DBN, LR-DBN Minus, CKT, and three variants of standard KT distinguished by how they update estimated skills: "full responsibility," "blame weakest, credit rest," and "update weakest subskill," with majority class as an additional baseline. Sections 4.1 and 4.2 describe our data and results.

**Table 1: Summary of models compared**

| Models | Fit | Predict | Update |
|---|---|---|---|
| **LR-DBN** | Train subskills together. Logistic regression assigns responsibility. | Logistic regression on subskill estimates. | Update subskills together. Logistic regression assigns responsibility. |
| **LR-DBN Minus** | Train subskills separately. Assign each one full responsibility. | | |
| **CKT** | | Multiply subskill estimates. | Update subskills together. Bayes equations assign responsibility. |
| **Full responsibility** | | | Update subskills separately, each with full responsibility. |
| **Blame weakest, credit rest** | | Minimum of subskill estimates. | |
| **Update weakest subskill** | | | Update only the weakest subskill. |
| **Majority class** | Identify larger class | Majority class | No update |

## 4.1 Data sets

We train and test the models on real data from two tutors used at schools. One data set is from children using Project LISTEN's Reading Tutor [9] at primary schools during the 2005-2006 school year. To model their oral reading fluency, we define performance $C^{(n)}$ as whether the Reading Tutor scored a text word as read fluently at step *n*, i.e., read without help or hesitation and recognized by the automated speech recognizer. We assume that whether a student read a word fluently depended on whether the student knew the requisite subskills, namely the grapheme-to-phoneme mappings in the word. Due to the large amount of data (1,792,103 read words from 275 students), we randomly selected 20 children who read a total of 80,268 words (3,972 distinct word types) with 320 unique grapheme-phoneme mappings. To counteract the prevalence of high-frequency words like *the*, we include at most the first 20 of each student's encounters of a word in the training data, leaving 24,145 read words. We do not limit the test data, so it includes 40,867 words.

The other data set [10] came from 123 high school students working on a geometry area unit of the Bridge to Algebra Cognitive Tutor®. The model for each student includes the same 50 subskills, and predicts whether the student will perform a step correctly. Again we include at most the first 20 of each student's encounters of a step in the training data, leaving 11,730 algebra practice steps, but 22,737 steps of test data.

The data sets from both tutors are unbalanced. The Reading Tutor scored 68.84% of the words in the training set as fluent, and 74.31% of the words in the test set. The Algebra Tutor rated 74.22% of the steps in the training data as correct, and 84.63% of the steps in the test data.

We fit each model separately for each student, as opposed to training a single model on the data for all the students. One reason is computational expedience: unlike methods that fit a separate model for each subskill, LR-DBN fits a single model for all the subskills, which involves processing much more data at a time. Training this model on all the students' data at once would be computationally unwieldy. The other reason is to compare methods fairly. Except for LR-DBN, it is feasible to train a single model of a subskill on the data for all the students, and in fact we tried it, but the resulting model does not perform as well as training a separate model for each student.

For all the methods, we fit the model for each student to the first half of the student's steps, and test it on the second half. We report average per-student accuracy on the unseen test data, weighting its mean and variance by per-student sample size to derive 95% confidence intervals. We use paired T-tests, paired by student, to rate LR-DBN's accuracy against each other method.

## 4.2 Results

Table 2 and Table 3 list all seven methods in decreasing order of their binary predictive accuracy on the test data. LR-DBN dramatically outpredicts all the other methods. LR-DBN's overall accuracy on the Reading Tutor data is 13% higher than majority class, vs. only 1% for the next method. For the Algebra Tutor data, LR-DBN is the only method that beats the majority class, by 7%. That is, on both data sets, LR-DBN has only half the error rate of the next best method.

For unbalanced data, accuracy on the minority class can be especially important. The minority class in our tutor data represents negative student outcomes to remediate by means of practice and instruction. LR-DBN beats every other method on the minority class by over 20% absolute in both data sets.

**Table 2: Mean per-student accuracy on Reading Tutor data** (95% confidence interval in parentheses) compared to LR-DBN is significantly (p<.01) worse if underlined, or *better if italicized*.

| Models | Accuracy | Accuracy Within Positive Class | Accuracy Within Negative Class |
|---|---|---|---|
| **LR-DBN** | 87.31% (±1.90%) | 91.17% (±2.80%) | 75.80% (±12.53%) |
| **Update weakest subskill** | 74.53% (±4.55%) | *95.06%* (±2.73%) | 15.15% (±5.29%) |
| **Majority class** | 74.31% | *100.00%* | 0.00% |
| **LR-DBN Minus** | 74.11% (±5.05%) | 90.71% (±7.89%) | 26.09% (±11.61%) |
| **Blame weakest, credit rest** | 73.90% (±4.59%) | 92.36% (±3.86%) | 20.52% (±6.43%) |
| **CKT** | 72.79% (±3.99%) | 89.47% (±3.52%) | 24.52% (±7.76%) |
| **Full responsibility** | 66.20% (±5.39%) | 72.30% (±10.24%) | 48.53% (±12.87%) |

**Table 3: Mean per-student accuracy on Algebra Tutor data** is significantly (p<.001) worse than LR-DBN's where underlined; *italicized values* are significantly better.

| Models | Accuracy | Accuracy Within Positive Class | Accuracy Within Negative Class |
|---|---|---|---|
| **LR-DBN** | 91.99% (±2.00%) | 96.5% (±1.30%) | 72.3% (±7.80%) |
| **Majority class** | 84.63% | *100.00%* | 0.00% |
| **CKT** | 84.38% (±1.14%) | *99.03%* (±0.26%) | 20.44% (±3.11%) |
| **Full responsibility** | 84.27% (±1.13%) | 95.65% (±0.88%) | 34.55% (±4.60%) |
| **LR-DBN Minus** | 83.92% (±1.17%) | 97.23% (±0.62%) | 25.80% (±3.84%) |
| **Blame weakest, credit rest** | 80.38% (±1.13%) | 90.70% (±0.72%) | 35.28% (±3.14%) |
| **Update weakest subskill** | 79.59% (±1.19%) | 91.13% (±0.69%) | 29.20% (±2.76%) |

What does comparison to LR-DBN Minus reveal about the relative contributions of the fitting and update procedures? LR-DBN Minus uses the same fitting procedure as conventional knowledge tracing, but uses logistic regression to update estimates. It performs substantially worse than LR-DBN, and comparably to the other methods. We conclude that LR-DBN's accuracy benefits more from its fitting procedure than from using logistic regression to combine estimates of hidden subskills.

Why does LR-DBN outpredict the other methods? Possible reasons include the strong assumptions that it avoids, but which they make implicitly by fitting and updating subskill estimates separately, multiplying them to predict performance on a step, and assigning each subskill full responsibility for the step's outcome. Inspection of Table 1 reveals that this last assumption is the only one they all have in common, implicating it as the likeliest culprit.

Predictive accuracy is just one way to evaluate student models. A more sensitive metric is model fit as measured by data likelihood, penalized by model complexity. Table 4 and Table 5 list the complexity-penalized model fits of the methods on the two data sets in increasing order, as scored by the Akaike information criterion (AIC) [11] and Bayesian information criterion (BIC) [12], defined respectively as:

$$AIC = 2k - 2\ln(L)$$

$$BIC = -2\ln(L) + k \cdot ln(n)$$

**Equation 15: Formulas for calculating AIC and BIC**

**Table 4: Complexity-adjusted Reading Tutor training data fit**

| Models | AIC | BIC | *k* |
|---|---|---|---|
| **LR-DBN** | 75,054.52 | 231,226.89 | 19,300 |
| **LR-DBN Minus** | 120,259.60 | 239,606.70 | 12,840 |
| **CKT** | 145,779.60 | 383,730.20 | 25,600 |
| **Full responsibility** | | | |
| **Blame weakest, credit rest** | | | |
| **Update weakest subskill** | | | |

**Table 5: Complexity-adjusted Algebra Tutor training data fit**

| Models | AIC | BIC | *k* |
|---|---|---|---|
| **LR-DBN** | 60,545.20 | 201,052.44 | 19,065 |
| **LR-DBN Minus** | 43,195.94 | 143,962.30 | 12,546 |
| **CKT** | 67,303.94 | 264,885.00 | 24,600 |
| **Full responsibility** | | | |
| **Blame weakest, credit rest** | | | |
| **Update weakest subskill** | | | |

Both AIC and BIC measure model fit as log-likelihood of the training data, *ln(L)*, penalized by model complexity (number of parameters, *k*). BIC also penalizes the number of observations, *n*. We calculate the number of parameters per student as follows:

LR-DBN fits the 3 groups of coefficients for each of the subskills and one intercept in Equation 10, plus two shared parameters, *guess* and *slip*. For the Reading Tutor data set, this number totals 3 × (320 + 1) + 2 = 965, multiplied by the 20 children in the data sample. For the Algebra Tutor data set, it totals 3 × (50 + 1) + 2 = 155, multiplied by 123 students.

LR-DBN Minus fits 2 parameters (*already know* and *learn*) per subskill, plus 2 shared parameters (*guess* and *slip*). This number of parameters per student totals (2 × 320) + 2 = 642 for the Reading Tutor and (2 × 50) + 2 = 102 for the Algebra Tutor.

The other methods fit 4 parameters (*already know*, *learn*, *guess* and *slip*) per subskill for each student, totaling 4 × 320 = 1280 for the Reading Tutor, and 4 × 50 = 200 for the Algebra Tutor.

Thus compared to previous methods, LR-DBN has about 1 less parameter per subskill, and LR-DBN Minus about 2 less.

What about the number *n* of observations? LR-DBN uses one observation per step to fit all the subskill parameters. In contrast, the other methods fit each subskill separately, assigning it full responsibility for every step that uses it, as if observing it separately for each subskill. Counting such duplicate observations as separate, they use three times as many Reading Tutor observations as LR-DBN, and twice as many Algebra Tutor observations.

Table 6 and Table 7 show the average log-likelihood of steps in the training and test data. All the methods except LR-DBN share the same likelihood on the training data because they fit parameters in the same way (as shown in Table 1).

**Table 6: Average log-likelihood for the Reading Tutor data**

| Models | On training data | On unseen test data |
|---|---|---|
| **LR-DBN** | -0.7549 | -0.3555 |
| **CKT** | -1.9586 | -1.1330 |
| **Full responsibility** | | -1.2230 |
| **Blame weakest, credit rest** | | -1.4944 |
| **LR-DBN Minus** | | -1.5690 |
| **Update weakest subskill** | | -1.6665 |

**Table 7: Average log-likelihood for the Algebra Tutor data**

| Models | On training data | On unseen test data |
|---|---|---|
| **LR-DBN** | -0.9555 | -0.1503 |
| **CKT** | -0.7717 | -0.2082 |
| **Full responsibility** | | -0.2065 |
| **Blame weakest, credit rest** | | -0.2529 |
| **LR-DBN Minus** | | -0.2364 |
| **Update weakest subskill** | | -0.2816 |

Normally one might expect log-likelihood to be lower for test data than training data, by an amount reflecting the degree of overfitting. However, the models assign higher likelihood to correct steps because, as Section 4.1 mentioned, they are more common than incorrect steps in the training data, and this difference is more pronounced in the test data. Its likelihood is therefore *higher*, and hence is not a direct gauge of overfitting.

Table 6 and Table 7 reveal that LR-DBN's log-likelihood is by far the highest on unseen test data from both tutors, consistent with how dramatically it outpredicts the other methods, even though they have higher log-likelihood on the training data from the Algebra Tutor. This reversal from training to test data suggests that the other methods might overfit that training data.

In summary, LR-DBN has a smaller number *k* of parameters than the other methods (except for LR-DBN Minus), a smaller number *n* of observations (counting duplicate observations as distinct), and higher likelihood on Reading Tutor training data, where it achieves the lowest AIC and BIC scores. Most important, LR-DBN far surpasses all the other methods in accuracy and log-likelihood on unseen test data from both tutors.

## 5. IMPLEMENTATION

To make LR-DBN publicly available[1], we added it to the Bayes Net Toolkit for Student Modeling (BNT-SM) [13]. BNT-SM inputs a data set and a DBN student model (not only the simple one used in standard knowledge tracing), specified in XML. It generates and executes BNT code to train and test the model, and outputs Excel files containing the parameter estimates and inference results. BNT is an open-source Matlab package[2] that supports many learning and inference algorithms for both static and dynamic Bayes models. BNT-SM hides most of the BNT coding details, freeing users to focus on constructing the student models rather than on programming them.

Using BNT-SM consists of four phases [13]:

1. Specify the data source in an XML specification.
2. Specify the DBN structure in XML.
3. Specify and initialize parameters in XML.
4. Call RunBnet.m in Matlab.

To fit LR-DBN on the Reading Tutor data with 320 subskills, we specify the structure shown in Figure 2 to BNT-SM in XML, as shown in the APPENDIX.

## 6. CONCLUSIONS

This paper makes multiple contributions to knowledge tracing:

First, we present a framework to characterize previous and new methods for tracing multiple subskills by how they (1) model knowledge tracing, (2) fit its parameters, (3) predict performance, and (4) update subskill estimates.

Second, we use data sets from reading and algebra tutors to compare LR-DBN against previous methods in terms of AIC, BIC, and predictive accuracy on unseen data, and show that LR-DBN performs significantly better on both data sets on all three metrics, cutting the best previous prediction error rate in half.

Third, we introduce the hybrid LR-DBN Minus method, which fits the same standard KT model as previous methods, but uses logistic regression to predict student performance.

Fourth, by comparing LR-DBN Minus to LR-DBN, we show that using logistic regression to predict performance suffices to beat previous methods, but that using logistic regression EM to jointly estimate subskills accounts for most of LR-DBN's superior performance.

Finally, in order to amplify the impact of this work, we have made LR-DBN publicly available and easy to extend to other student modeling with dynamic Bayes nets, by incorporating it into the latest version of the BNT-SM student modeling toolkit [13] used in previous studies of knowledge tracing [e.g., 14].

This work has several limitations for future work to address.

First, LR-DBN has so far been applied just to simple knowledge tracing of multiple subskills, but it can apply to any DBN. Future work could use LR-DBN to improve other DBN student models, for example to measure more accurately the scaffolding and learning effects of tutor help [14].

Second, LR-DBN needs 5.5 hours on average per student to fit and update; the other methods take less than 1 hour to fit a single set of parameters for all the students and subskills, and 2-5 minutes to update. Future work may train LR-DBN faster or develop other methods that are faster to train. Such work might adapt two previous types of cognitive diagnosis models that operate on static data and have statistical learning algorithms, both EM and MCMC [15]. NIDA (Noisy Inputs, Deterministic "And" gate) models [16] resemble CKT because it applies *guess* and *slip* to individual subskills before combining them conjunctively. DINA (Deterministic Inputs, Noisy "And" gate) models [17] resemble LR-DBN because it combines subskills (with logistic regression) before applying *guess* and *slip* to the resulting knowledge state. Extending either type of model to apply to knowledge tracing may improve LR-DBN itself.

Finally, although LR-DBN traces multiple subskills better than previous methods, it (like them) must be told which steps use which subskills. Future work may infer this information automatically [18].

## ACKNOWLEDGMENTS

This work was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080628 to Carnegie Mellon University. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or U.S. Department of Education. We thank Ken Koedinger for his CKT implementation and algebra tutor data, and children, schools, and LISTENers for our Reading Tutor data.

[1] At http://www.cs.cmu.edu/~listen/BNT-SM

[2] At http://code.google.com/p/bnt

## APPENDIX

To use LR-DBN in BNT-SM, we first specify its data source:

```
<multi_subskill>yes</multi_subskill>
<input>
   <evidence_train>evidence.train.xls</evidence_train>
   <evidence_test>evidence.test.xls</evidence_test>
</input>
<output>
   <param_table>param_table.xls</param_table>
   <inference_result>inference_result.xls</inference_result>
   <inference_result_header>inference_result.xls</inference_re
sult_header>
   <log>log.txt</log>
</output>
```

To add logistic regression to standard knowledge tracing, we represent the ***320*** subskills as a single ***multi*** node ***kc***, which transits to the latent node ***knowledge*** within a step. The hidden state of ***knowledge*** transits both to the output ***fluent*** within the current step and to the ***knowledge*** state at the next step:

```
<nodes>
   <node>
         <id>1</id>
         <name>kc</name>
         <type>multi</type>
         <values>320</values>
         <latent>no</latent>
         <prefix_field>kc</prefix_field>
         <within>
               <transition>knowledge</transition>
         </within>
         <between></between>
   </node>
   <node>
         <id>2</id>
         <name>knowledge</name>
         <type>discrete</type>
         <values>2</values>
         <latent>yes</latent>
         <field> knowledge</field>
         <within>
         <transition>fluent</transition>
         </within>
         <between>
         <transition>knowledge</transition>
         </between>
   </node>
   <node>
         <id>3</id>
         <name>fluent</name>
         <type>discrete</type>
         <values>2</values>
         <latent>no</latent>
         <field>fluent</field>
         <within></within>
         <between></between>
   </node>
</nodes>
```

Then we define and set initial values of the LR-DBN parameters. We specify the input node ***kc*** as ***root*** to have no parents and no parameters, the latent node ***knowledge*** as ***softmax*** to have a multinomial logit function, and the output node ***fluent*** to have a simple ***discrete*** conditional probability table, with random initial parameter values in LR-DBN's EM fitting algorithm:

```
<eclasses>
   <eclass>
         <id>1</id>
         <formula>P1(kc)</formula>
         <type>root</type>
   </eclass>
   <eclass>
         <id>2</id>
         <formula>P2(knowledge </formula>
         <type>softmax</type>
         <cpd>
               <eq>P2(T)</eq>
               <init>rand</init>
               <param>L0</param>
               <eq>P2(F)</eq>
               <init>1-P1(T)</init>
               <param>null</param>
         </cpd>
   </eclass>
   <eclass>
         <id>3</id>
         <formula>P3(fluent| knowledge </formula>
         <type>discrete</type>
         <cpd>
               <eq>P3(T|F</eq>
               <init>rand</init>
               <param>guess</param>
               <eq>P3(F|T)</eq>
               <init>rand</init>
               <param>slip</param>
               <eq>P3(F|F)</eq>
               <init>1-P3(T|F)</init>
               <param>null</param>
               <eq>P3(T|T)</eq>
               <init>1-P3(F|T)</init>
               <param>null</param>
         </cpd>
   </eclass>
   <eclass>
         <id>4</id>
         <formula>P4(knowledge| knowledge)</formula>
         <type>softmax</type>
         <cpd>
               <eq>P4(T|F </eq>
               <init>rand</init>
               <param>learn</param>
               <eq>P4(F|T)</eq>
               <init>rand</init>
               <param>forget</param>
               <eq>P4(F|F </eq>
               <init>1-P4(T|F)</init>
               <param>null</param>
               <eq>P4(T|T)</eq>
               <init>1-P4(F|T)</init>
               <param>null</param>
         </cpd>
   </eclass>
</eclasses>
```

## REFERENCES


[1] Corbett, A. and J. Anderson, *Knowledge tracing: Modeling the acquisition of procedural knowledge.* User modeling and user-adapted interaction, 1995. **4**: p. 253-278.

[2] Xu, Y. and J. Mostow. *Using Logistic Regression to Trace Multiple Subskills in a Dynamic Bayes Net*. in *Proceedings of the 4th International Conference on Educational Data Mining* 2011. Eindhoven, Netherlands.

[3] Cen, H., K. Koedinger, and B. Junker. *Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement*. in *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. 2006. Jhongli, Taiwan.

[4] Cen, H., K.R. Koedinger, and B. Junker. *Comparing Two IRT Models for Conjunctive Skills*. in *Ninth International Conference on Intelligent Tutoring Systems*. 2008. Montreal.

[5] Gong, Y., J. Beck, and N.T. Heffernan. *Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting Procedures*. in *Proceedings of the 10th International Conference on Intelligent Tutoring Systems*. 2010. Pittsburgh, PA: Springer Berlin / Heidelberg.

[6] Koedinger, K.R., et al., *Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing*, in *Proceedings of the 4th International Conference on Educational Data Mining*. 2011: Eindhoven, NL. p. 91-100.

[7] Dempster, A., N. Laird, and D. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society Series B (Methodological), 1977. **39(1)**: p. 1-38.

[8] Xu, Y. and J. Mostow. *Logistic Regression in a Dynamic Bayes Net Models Multiple Subskills Better! [Best Poster Nominee]*. in *Proceedings of the 4th International Conference on Educational Data Mining* 2011. Eindhoven, Netherlands.

[9] Mostow, J. and G. Aist, *Evaluating tutors that listen: An overview of Project LISTEN*, in *Smart Machines in Education*, K. Forbus and P. Feltovich, Editors. 2001, MIT/AAAI Press: Menlo Park, CA. p. 169-234.

[10] Koedinger, K.R., et al., *A Data Repository for the EDM community: The PSLC DataShop*, in *Handbook of Educational Data Mining*, C. Romero, et al., Editors. 2010, CRC Press: Boca Raton, FL. p. 43-55.

[11] Akaike, H., *A new look at the statistical model identification*. IEEE Transactions on Automatic Control, 1974. **19**(6): p. 716–723.

[12] McQuarrie, A.D.R. and C.-L. Tsai, *Regression and Time Series Model Selection*. 1998: World Scientific.

[13] Chang, K.-m., et al., *A Bayes Net Toolkit for Student Modeling in Intelligent Tutoring Systems*, in *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, K. Ashley and M. Ikeda, Editors. 2006: Jhongli, Taiwan. p. 104-113.

[14] Beck, J.E., et al. *Does help help? Introducing the Bayesian Evaluation and Assessment methodology*. in *9th International Conference on Intelligent Tutoring Systems*. 2008. Montreal.

[15] de la Torre, J., *DINA Model and Parameter Estimation: A Didactic.* Journal of Educational and Behavioral Statistics, 2009. **34**(1): p. 115-130.

[16] Hartz, S., *A Bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality*. 2002, University of Illinois at Urbana-Champaign: Unpublished doctoral dissertation.

[17] Torre, J.d.l. and J. Douglas, *Higher-order latent trait models for cognitive diagnosis.* Psychometrika, 2004. **69**: p. 333-353.

[18] González-Brenes, J.P. and J. Mostow. *Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models*. in *Proceedings of the Fifth International Conference on Educational Data Mining*. 2012, in press. Chania, Crete, Greece.

# Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models

José P. González-Brenes and Jack Mostow
Project LISTEN
Language Technologies Institute
Carnegie Mellon University
{joseg, mostow}@cs.cmu.edu

## ABSTRACT

This work describes a unified approach to two problems previously addressed separately in Intelligent Tutoring Systems: (i) Cognitive Modeling, which factorizes problem solving steps into the latent set of skills required to perform them [7]; and (ii) Student Modeling, which infers students' learning by observing student performance [9].

The practical importance of improving understanding of how students learn is to build better intelligent tutors [8]. The expected advantages of our integrated approach include (i) more accurate prediction of a student's future performance, and (ii) clustering items into skills automatically, without expensive manual expert knowledge annotation.

We introduce a unified model, Dynamic Cognitive Tracing, to explain student learning in terms of skill mastery over time, by learning the Cognitive Model and the Student Model jointly. We formulate our approach as a graphical model, and we validate it using sixty different synthetic datasets. Dynamic Cognitive Tracing significantly outperforms single-skill Knowledge Tracing on predicting future student performance.

## 1. INTRODUCTION

We propose Dynamic Cognitive Tracing as a method that estimates from performance data:

1. **A Student model.** The estimate of a student's knowledge of a skill in a given time.
2. **A Cognitive Model.** The skills a students require to solve a problem step.

Let's illustrate the student modeling problem with an example. Suppose we are interested in modeling data from a reading tutor that listens to children read aloud. Figure 1 shows sample data in this scenario. We follow the convention of referring to the scorable steps in an intelligent tutor task as "items" [27]. The input variable is the item $id_t$, which in this case is the word read by a student at time step $t$. The target variable $p_t$ is the performance of the student– in this case whether the tutor accepted the word read. The student reads the words "smile because it" correctly, but misreads the word "happened". The student modeling problem is to predict future student performance.

Existing student modeling techniques require cognitive models, assignments of items to skills [9]. This is a very expensive requirement, since it often depends on expert domain knowledge [4]. For example, in our reading tutor scenario, it is not a trivial endeavor to cluster a dictionary of words into the set of skills needed to read them.

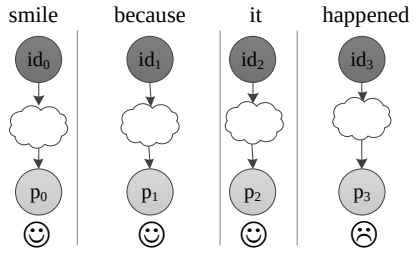



Figure 1: Reading tutor example of student modeling

Unfortunately, the success of existing methods for automatic construction of cognitive models has been limited [11]. Current methods for discovering cognitive models are restricted in that they cannot handle longitudinal data, or that they are not fully automatic. For example, Principal Component Analysis, Non-Negative Matrix Factorization [27] and the Q-Matrix Method [2] ignore the temporal dimension of the data. On the other hand, Learning Factors Analysis [7] is designed for temporal data, but it requires an expert's cognitive model. Our main contribution is a fully automatic approach to discover a cognitive model of longitudinal student data. Our goal is discovering student models, while simultaneously clustering similar items together.

The rest of this document is organized as follows. Section 2 reviews related prior work. Section 3 describes our approach, Dynamic Cognitive Tracing, to jointly learn a student model jointly with a factorization of items into skills. Section 4 evaluates performance using synthetic data. Section 5 provides some concluding remarks.

## 2. RELATION TO PRIOR WORK

In this section we study Dynamic Cognitive Tracing's relation with prior work. Section 2.1 surveys previous approaches to learn student models. Section 2.2 summarizes automatic approaches for cognitive model discovery.

## 2.1 Student Modeling

Corbett and Anderson [9]'s seminal paper introduced Knowledge Tracing as a way to model students' changing knowledge during skill acquisition. It uses (a) a cognitive model that maps a problem solving item to the skills required, and (b) logs of students' correct and incorrect answers as evidence of their knowledge on a particular skill. Reye [22] showed that there is an equivalent formulation of Knowledge Tracing as a Bayesian Network. Knowledge Tracing has enabled significantly faster teaching by Intelligent Tutors, while achieving the same performance on evaluations [8].

Knowledge Tracing, as well as Dynamic Cognitive Tracing, are non-convex problems. This means that the optimizer that estimates the parameters of the models might get stuck in local optima far away from the global optimum. Moreover, these formulations are also non-identifiable: There exist potentially many student models that may explain the data observed equally-well. In Knowledge Tracing, the main source of non-identifiability is the trade-off between the probability of a student's initial knowledge, and the probability of learning the skill [5]. To mitigate non-identifiability, recent work has proposed the use of Bayesian priors [5] or using contextual clues to estimate whether a student has guessed [1].

Other approaches to student modeling include Performance Factor Analysis [19, 14], which predicts student performance based on the item difficulty and student historical performances. Alternatively, Learning Decomposition [6], uses non-linear regression to determine how to weight different types of practice opportunities relative to each other. More recently, Tensor Factorization [25], has been used to the student modeling problem. It use recommender system techniques to learn student models. None of these techniques aim to discover cognitive models. Thai-Nghe et al. [25] make use of latent variables, but they argue that it is not possible to interpret their semantics. Their formulation is tied to specific students, and it is not clear how to generalize their approach to unseen students in the training set, or when students encounter only a very sparse set of items. We designed Dynamic Cognitive Tracing aiming to discover latent factors with the interpretation of Cognitive and Student Models.

Desmarais [11] argues that the construction of a cognitive model from data is highly desirable, not only to avoid the labor intensive task of specifying which skills are involved in which task, but because a data-driven approach might outperform human judgment. In the next subsection we study such approaches.

## 2.2 Automatic Discovery of Cognitive Models

Winters et al. [27] surveyed methods for automatic construction of cognitive models. Examples are matrix factorization techniques, such as Principal Component Analysis (PCA) and Non-Negative Matrix Factorization (NNMF). The theoretical relationships between different matrix factorization techniques has been studied in detail [24].

The Q-matrix algorithm [2, 3], is a hill-climbing method that creates a cognitive model linking skills and items directly from student response data. An alternative approach, Learning Factors Analysis [7], performs combinatorial search to evaluate and improve on existing cognitive models.

None of the techniques reviewed in this section take into account the temporal dimension of the data without human intervention. To the extent of our knowledge, we are the first ones to estimate a cognitive model completely automatically from data collected over time.

# 3. DYNAMIC COGNITIVE TRACING

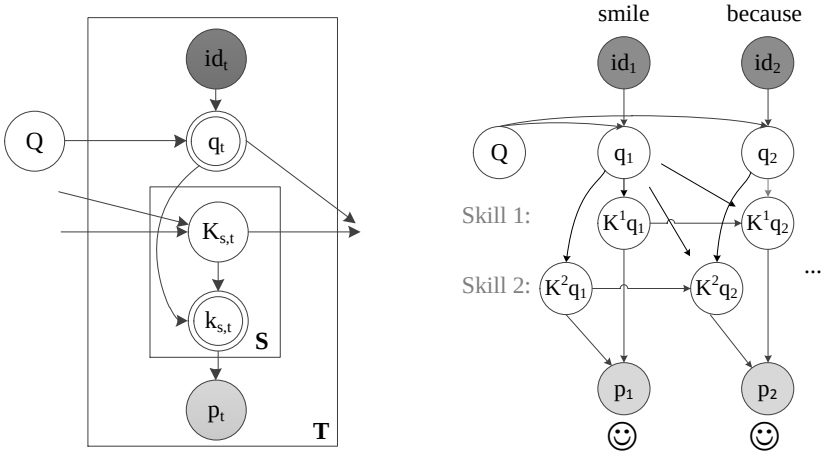
We now describe Dynamic Cognitive Tracing. Subsection 3.1 details our approach. Subsection 3.2 provides pointers on the training and inference algorithms used. Subsection 3.3 shows how Dynamic Cognitive Tracing relates two common techniques used in student modeling and in automatic generation of a cognitive model.

## 3.1 Model

We formulate Dynamic Cognitive Tracing as a Bayesian Network. Bayesian Networks [20], are a popular framework to reason using noisy information. Bayesian networks are directed acyclic graphical models where the nodes are variables and the edges specify statistical dependencies between variables.

Bayesian Networks are often described using plate diagram notation to show the statistical relationship between their random variables. The plate diagram of Dynamic Cognitive Tracing is shown in Figure 2(a). Instead of drawing a variable multiple times, we follow the convention of using a plate to group repeated variables. As an example, we unroll Dynamic Cognitive Tracing using two skills in Figure 2(b). The description of the generative story of the variables is described in Figure 3. We follow the convention of using dark-gray to color variables that are observable during both training and testing. Variables visible during testing only are colored in light gray. Latent variables, which are never observed, are denoted in white circles. The double-line around variables is used to indicate that their value is calculated deterministically given its parents. The variables in Dynamic Cognitive Tracing are:

- **S** is the number of skills in the model.
- **Ids** is the number of items that the student can practice with the tutor. For example, in the case of a reading tutor, **Ids** is the vocabulary size. If the tutor is creating items on the fly, **Ids** is the number of templates from where items are being generated.
- $Q$ is an $\mathbf{Id} \times \mathbf{S}$ matrix that maps items to skills. Each row $Q_{id}$ is modeled as a multinomial representing the skills required for item $id$. For example, if $Q_{id_t} = [0.5, 0.5, 0, 0]$, we interpret item $id_t$ to be a mixture of skills 1 and 2. In this example $id_t$ does not require skills 3 and 4. $Q$ need not be hidden. If in fact $Q$ is known, we can clamp the parameters to their known values.
- $q_t$ is the skill for item $id_t$. For example, $q_t = 1$ iff skill 1 is required for item $id_t$, $q_t = 2$ iff skill 2 is required, and so on. $q_t$ is chosen deterministically as the row number $id_t$ of $Q$.
- $K_{s,t}$ indicates whether the student has the knowledge of skill $s$. Notice, there is a markovian dependency across time steps: if skill $s$ is known at time $t-1$, it is likely to be known it at time $t$. Therefore, we also need to know which skills were active on the previous time step (i.e., $k_{s,t}$ depends on $q_{t-1}$). For simplicity, in this




(a) Plate diagram

(b) Unrolled example with two skills

Figure 2: Dynamic Cognitive Tracing as a graphical model

work we treat each $K$ as a binary variable (whether the skill is known or not).

- $k_{s,t}$ is a binary variable that represents if the skill is known and required by the item $id_t$. Hence, its value is computed deterministically by applying a dot product to its parents: $k_{s,t}$ is true iff skill $s$ is required ($q_t = s$), and the student has learned the skill ($K_{s,t} = 1$).
- $p_t$ is the target variable that models performance. It is only observed during training.
  - For discrete grades (i.e., right or wrong), a Binomial distribution or logistic regression can be used. The use of logistic regression in Bayesian Networks has been studied in the context of mixture of experts [16], and more recently for the multiple subskill problem in student modeling [28]. In this paper we use the Binomial approach.
  - For continuous grades, (i.e., $0 \sim 100$) linear regression can be used.

Our main contribution is unsupervised estimation of the cognitive model $Q$ from longitudinal data, while simultaneously estimating the student model parameters. In the next subsection we study how to learn the parameters of Dynamic Cognitive Tracing, as well as how to perform inference on it.

## 3.2 Training and Inference

Dynamic Cognitive Tracing is formulated as a directed graphical model (Bayesian Network). We leverage existing technologies to quickly implement a prototype of Dynamic Cognitive Tracing. We used the Bayesian Network Toolkit [18] (BNT) for Matlab.

As described in the previous subsection, the knowledge of a skill is dependent of its value on the previous time step. This kind of dependency is called a Markov Chain. Therefore, in Dynamic Cognitive Tracing, the student knowledge

1. Draw $Q_{id} \sim$ Multinomial; **Ids** times
2. For each time-step $t \in \{0 \dots T\}$:
   (a) Draw $id_t \sim$ Multinomial
   (b) For each skill $s \in \{0 \dots \mathbf{S}\}$ :
   (c) Set $q_{s,t} \leftarrow Q_{id_t}$
   (d) Draw $K_{s,t} \sim$ Binomial
   (e) Set $k_{s,t} \leftarrow K_{q_t} \cdot q_{s,t}$
   (f) $p_t \sim \mathcal{N}(k_{1,t}, k_{2,t}, \dots, k_{S,t})$, for continuous $p$, or for binary variables either $p_t \sim \mathit{logistic}(k_{1,t}, k_{2,t}, \dots, k_{S,t})$, or $p_t \sim$ Binomial

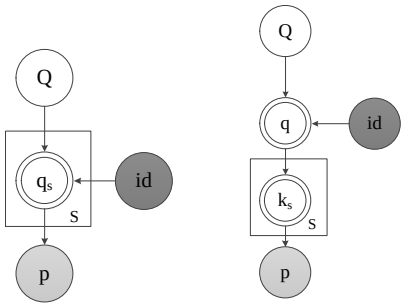
Figure 3: Generative story of Dynamic Cognitive Tracing

of $S$ skills is modeled using $S$ layers of Markov Chains. Unfortunately, this is not scalable, because exact inference on layers of Markov Chains that produce a single output is untractable: the runtime complexity grows exponentially on the number of layers [12]. Hence, we limit our study to a small number of skills. In future work we will implement inference techniques that scale better, like Gibbs Sampling.

The name Bayesian Network is a misnomer, because it does not require to use Bayesian Estimation, as in fact, we used Maximum Likelihood Estimation to perform exact inference. BNT implements the Junction Tree algorithm [15], an inference algorithm that generalizes the the Forward-Backward algorithm that is used in Knowledge Tracing and Hidden Markov Models [21]. To estimate the parameters of the model, we use the Expectation-Maximization (E-M) algorithm [10]. Like all non-convex optimizaters, E-M is not guaranteed to find the globally optimal solution.

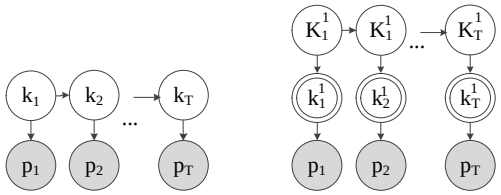
## 3.3 Unifying Perspective

We now discuss how Dynamic Cognitive Tracing generalizes two common techniques for cognitive and student modeling.




(a) PPCA (b) Dynamic Cognitive Tracing

Figure 4: Two-skill models with one time step




(a) Knowledge Tracing (b) Dynamic Cognitive Tracing

Figure 5: Unrolled graphical model representation of one-skill student models

Cognitive models have been built by matrix factorization techniques [27]. Probabilistic Principal Component Analysis (PPCA) [26] is an example of such matrix factorization techniques. It is a formulation of the Principal Component Analysis algorithm using graphical models. The main advantages of this approach over conventional PCA, is that it can handle missing data, and it provides a probabilistic interpretation of the underlying factors.

In Figure 4(a) we show the graphical model representation of PPCA when explicitly formulated to handle missing data. If the variable $p$ is continuous, it is modeled with a Gaussian. If the variable $p$ is discrete, it is model with a Binomial, using a logistic link function. Discrete PCA is also known in the literature as Logistic PCA [23]. Figure 4(b) shows the simplified Dynamic Cognitive Tracing with two skills, when there is no temporal information available. The structure of both graphical models is very similar: in both cases, the performance is explained by latent variables that represent the skills. The main difference is that Dynamic Cognitive Tracing takes into account the knowledge of the skill estimated from the student model: the performance is explained by the latent knowledge of the skills. We hypothesize that the advantage of our approach lies in the fact that it is not limited to a single timestep like PPCA is. We expect that item-performance data to be very noisy, and that the temporal information would be useful to model skill acquisition.

Figure 5(a) shows the graphical model representation of Knowlege Tracing with a single skill model, which is just a Hidden Markov Model. Figure 5(b) shows the unrolled single-skill Dynamic Cognitive Tracing ($\mathbf{S} = 1$) counterpart. In this case the structure of Dynamic Cognitive Tracing is equivalent to Knowledge Tracing.

# 4. EMPIRICAL EVALUATION

In this section, we report results of using Dynamic Cognitive Tracing to predict future student performance using synthetically generated datasets. In the context of this paper, we decouple the problem of discovering the assignments of items to skills and the problem of discovering the number of skills. For our experiments, we assume the number of skills is known. In a real scenario, where the number of skills is unknown, it could be estimated by using cross-validation using a held-out set. We report our results using Dynamic Cognitive Tracing using the true number of skills.

Dynamic Cognitive Tracing aims to discover the skills automatically without supervision. We compare if the cognitive model estimated by Dynamic Cognitive Tracing outperforms a cognitive model that assigns all of the items to a single skill. Therefore, as a baseline, we compare against Knowledge Tracing using a single skill.

In all comparisons between Knowledge Tracing and Dynamic Cognitive Tracing, their parameters are estimated using the same training set. The testing and training sets do not overlap students.

## 4.1 Experimental setup

In this section, we describe the synthetic data sets generation criteria and the evaluation metrics. To generate the synthetic data sets, we use the generative story described in Figure 3, having each student encounter 25 items during training (sequence length = 25). In preliminary experiments, we noticed that by the $25^{\text{th}}$ time step, most synthetic students learned. To have a more balanced test set that has roughly the same number of correct and incorrect answers, the sequence length of the test set is sampled randomly.

We want synthetic data to be plausible; for example, the probability of answering an item correctly by guessing should be lower than the probability of answering an item correctly due to knowledge. Therefore, the synthetic datasets follow these constraints:

- The *learning probability*, the probability of transitioning from not knowing a skill, to knowing it, lies in $[0.01 \ldots 0.45]$.
- The *guess probability*, the probability of answering correctly, given that the student does not know the skill, lies in $[0.01 \ldots 0.30]$.
- The *slip probability*, the probability of answering incorrectly, given that the student knows the skill, lies in $[0.01 \ldots 0.30]$.

Note that these constraints are only exercised for generating the data. None of our models make use of this prior knowledge. For simplicity, in this paper we limit studying cognitive models that have only one skill active per item, but Dynamic Cognitive Tracing does not make use of this information. We constrain the models to not learn the "forget probability" (e.g., the transition probability from "knowing" to "not knowing" is zero).

Knowledge Tracing can sometimes provide bad parameter estimates. Beck and Chang [5] argued that when Knowledge Tracing performs badly, it is often because of incorrect estimation of the initial knowledge of the students (initial probabilities). We want to make sure that our results are better than Knowledge Tracing because of the strengths of Dynamic Cognitive Tracing, not because Knowledge Tracing got stuck in an "unlucky" local optimum. Therefore, we constrain all of the students to not have any initial knowledge in our experiments.

E-M is used to learn the parameters of the models. Knowledge Tracing and Dynamic Cognitive Tracing are initialized with random parameters, however, the emission probabilities (slip and guess probabilities) of Dynamic Cognitive Tracing are initialized using a single-skill model. We experiment running E-M using five different random initializations.

Unless noted otherwise, each dataset is divided in three parts: (i) a training set with 200 students, (ii) a development set with 50 students, used to choose the best out of five random initializations of the E-M algorithm, and (iii) a test set with 50 students. Students do not overlap among the sets.

We report the performance of our models using two metrics:

- **Average Per-item Likelihood.** Likelihood is a common metric to evaluate models that find latent structure [12]. It measures how likely a model is to predict the test set. It penalizes more heavily incorrect predictions with high-confidence. More formally, let $I$ be the number of students in the test set, let $\hat{p}_{i,t}$ be the estimated performance of student $i$ at time $t$, let $p_{i,t}$ be the real performance of the student and let $T_i$ be the number of time steps for student $i$. Then we compute the per-item likelihood as:

$$\frac{\sum_{i}^{I}\sum_{t}^{T_i} \text{pr}(\hat{p}_{i,t} = p_{i,t} | \hat{p}_{i,t-1}, id_{i,t})}{\sum_{i}^{I} T_i}$$

- **Classification Accuracy.** Classification accuracy measures how often the predicted performance matches the actual performance. Formally, let $\delta(\cdot)$ be the Indicator function that returns 1 iff its argument is true, and 0 otherwise. We compute the accuracy as:

$$\frac{\sum_{i}^{I}\sum_{t}^{T_i} \delta(\text{pr}(\hat{p}_{i,t} = p_{i,t} | \hat{p}_{i,t-1}, id_{i,t}) > 0.5}{\sum_{i}^{I} T_i}$$

In the next section, we report all of the different parameter combinations of parameters we used to experiment. We did not perform any additional tuning besides the one reported in the next section.

## 4.2 Results

We create a total of 60 random synthetic datasets using the constraints explained in Section 4.1. All of them have four types of items (**Ids** = 4). We created twenty datasets with 2, 3 and 4 skills (**S** = 2, 3, 4), respectively.

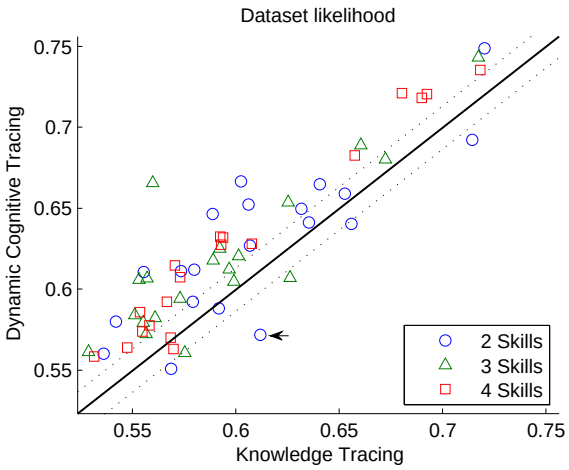



Figure 6: Average Likelihood of Dynamic Cognitive Tracing and single-skill Knowledge Tracing in 60 different data sets

Table 1: Dynamic Cognitive Tracing's worst performing dataset (highlighted in Figure 6)

| | Skill 1 | Skill 2 |
|---|---|---|
| Learning probability: | .35 | .30 |
| Slip probability: | .09 | .08 |
| Guess probability: | .02 | .11 |

In Figure 6, the horizontal axis denotes the Likelihood of single-skill Knowledge Tracing. The vertical axis is the Likelihood of Dynamic Cognitive Tracing. The solid line divides the datasets in which Dynamic Cognitive Tracing performed better than Knowledge Tracing (upper left corner) and the ones in which it performed worse (lower right corner). The dotted lines represent the confidence interval for the mean of the Likelihood of Knowledge Tracing. Dynamic Cognitive Tracing performs as well or above the baseline in a total of 52 (87%) of the datasets.

Is estimating a cognitive model with Dynamic Cognitive Tracing better than assuming a single skill model? We compare the mean Likelihood of Dynamic Cognitive Tracing ($\bar{x}_{\text{DCT}} = 62.34, s_{\text{DCT}} = 5.13$), with the mean Likelihood of single-skill Knowledge Tracing ($\bar{x}_{\text{KT}} = 59.97, s_{\text{KT}} = 5.18$). The null hypothesis is that the mean Likelihood of both models is the same ($H_0 : \mu_{DCT} = \mu_{KT}$). We perform a two-tailed $t$-test, pairing on the datasets (n=60). We reject the null hypothesis $H_0$ with confidence $p < 0.05$. We conclude that Dynamic Cognitive Tracing outperforms Knowledge Tracing with a single skill assumption.

In Figure 6 the arrow points to the dataset that performs the worst compared to the single-skill Knowledge Tracing baseline. The Likelihood of the true model is 65%, of Dynamic Cognitive Tracing is 57%, and of single-skill Knowledge Tracing is 61%. We now investigate why Knowledge Tracing outperforms Dynamic Cognitive Tracing on this specific dataset. Table 1 shows the parameters of the student model. We notice that both skills' learning and slip probabilities are very similar. We run the E-M algorithm using 100 different random initializations for both Dynamic Cog-

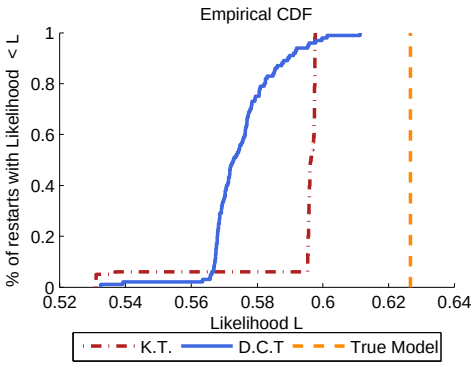



Figure 7: Cumulative Distribution Function of the Likelihood over 100 restarts (using the dataset highlighted in Figure 6)

Table 2: Model Comparison Over Number of Skills

| | 2 skills | | 3 skills | | 4 skills | |
|---|---|---|---|---|---|---|
| | Acc. | Lik. | Acc. | Lik. | Acc. | Lik. |
| True model | .75 | .64 | .75 | .61 | .76 | .62 |
| DCT | .74 | .63 | .73 | .62 | .73 | .62 |
| KT(1 skill) | .71 | .61 | .69 | .59 | .70 | .60 |
| Majority | .63 | - | .66 | - | .67 | - |

nitive Tracing and Knowledge Tracing. We use the same training set used for the highlighted dataset of Figure 6. To ensure more reliable results, we use a larger test set of 200 students (instead of 50 students). Figure 7 shows the Cumulative Distribution Function of the Likelihood over 100 random initializations. For a specific Likelihood $\ell$ in the horizontal axis, the vertical axis is the percentage of initializations with Likelihood found at a value less than or equal to $\ell$. Figure 7 shows that the Likelihood of the true model is 62.6%. The best Likelihood of Dynamic Cognitive Tracing is 61.1%, and of single-skill Knowledge Tracing is 59.7%. Knowledge Tracing gets stuck in local optima in less than 5% of the restarts. On the other hand, for this dataset, Dynamic Cognitive Tracing gets stuck in local optima 99% of the time. While there is a Dynamic Cognitive Tracing solution that outperforms Knowledge Tracing, the E-M algorithm found it in 4% of the initializations.

In Table 2, we aggregate the results of Figure 6. We report the mean performance of the parameters that generate the 60 synthetic data sets (True model), Dynamic Cognitive Tracing, single-skilled Knowledge Tracing (KT), and the classifier that always predicts the majority class (Majority). We present the mean Classification Accuracy and the mean Likelihood. Dynamic Cognitive Tracing has a similar Likelihood and Classification Accuracy to the True Model and dominates Knowledge Tracing.

Let's study a sample cognitive model estimated using Dynamic Cognitive Tracing. Here $Q^*$ is the True Model's cognitive model from which the synthetic data was generated. An estimate $\hat{Q}$, learned from data using our approach is:

$$Q^* = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \hat{Q} = \begin{pmatrix} 0.85 & 0.15 \\ 0.43 & 0.57 \\ 0.27 & 0.73 \\ 0.91 & 0.09 \end{pmatrix}$$

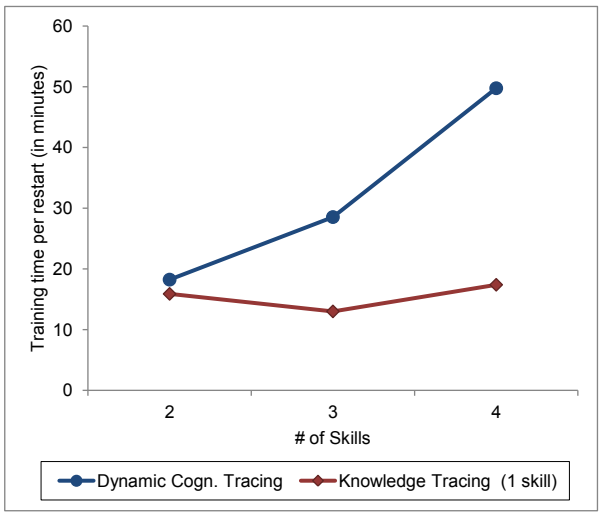



Figure 8: Time required (in mins.) to train a single restart

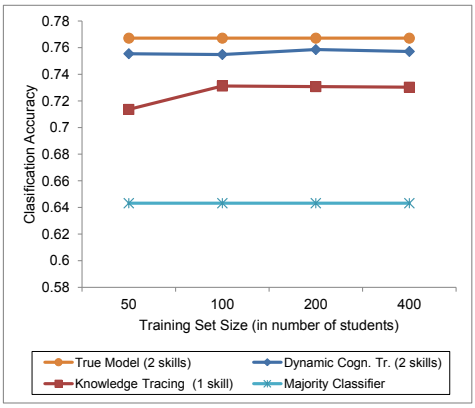



Figure 9: Classification accuracy using different training set sizes

The estimated cognitive model has some uncertainty, but if we round $\hat{Q}$ to integer values, it matches $Q^*$. In future work, we are interested in using Bayesian priors to encourage sparse entries in $\hat{Q}$ [13]. Bayesian estimation is not currently supported by the BNT toolkit in which we implemented our model.

In Figure 8 we show how long it took to perform a single restart of Dynamic Cognitive Tracing and Knowledge Tracing. Although Dynamic Cognitive Tracing achieves better accuracy, its exact inference implementation does not scale well with the number of skills.

We now try to simulate the effect of different amount of training data. For this, we experiment with 50, 100, 200 and 400 students. We observed that in the PSLC DataShop [17], a repository for student data sets, it is common for smaller datasets to have data from at least 50 students. We assess the performance of our approach using ten synthetic training sets with different number of students. For all experiments here, we used four different types of items (**Ids** = 4), and two skills (**S** = 2). In Figure 9, the "True model" line represents the classification accuracy of the model using the parameters from where the synthetic data was generated. The Knowledge Tracing line shows the performance of this

Table 3: Model Comparison Over Number of Items

| | **Ids** = 4 | | **Ids** = 8 | | **Ids**=12 | |
|---|---|---|---|---|---|---|
| | Acc. | Lik. | Acc. | Lik. | Acc. | Lik. |
| True model | .77 | .66 | .70 | .60 | .73 | .61 |
| DCT | .76 | .65 | .67 | .58 | .66 | .57 |
| KT(1 skill) | .73 | .63 | .64 | .56 | .66 | .57 |
| Majority | .66 | - | .59 | - | .58 | - |

approach, using a single skill. The results suggest that the approaches compared can achieve good performance even on a smaller datasets.

Since we are actually clustering similar items into skills, the number of different items (**Ids**) may have an impact on the performance of our approach. We create ten sets with 4, 8 and 16 item types respectively ($\mathbf{Id} = 4, 8, 16$). All of them have two skills ($\mathbf{S} = 2$). In Table 3, we summarize the Likelihood and the Classification Accuracy of different models. The true model's parameters achieve the highest likelihood, followed by our approach, that dominates Knowledge Tracing.

# 5. CONCLUSION

We propose Dynamic Cognitive Tracing as a novel unified approach to two problems previously addressed separately in Intelligent Tutoring Systems: (i) Student Modeling, which infers students' learning by observing student performance [9], and (ii) Cognitive Modeling, which factorizes problem solving steps into the latent set of skills required to perform them [7].

We provide empirical results using synthetic data supporting that our unsupervised approach is better than assuming that all items come from the same skills. Dynamic Cognitive Tracing significantly outperforms Knowledge Tracing using a single skill assumption.

We used the Bayesian Networks Toolkit to quickly prototype our approach. However, our prototype is limited in that (i) the inference algorithm used by the toolkit leads to complexity exponential in the number of skills, and (ii) the optimization algorithm gets stuck in local optima. We recommend implementing Dynamic Cognitive Tracing using approximate inference as future work.

For simplicity, in this paper we limited our study to synthetic data of items that require a single skill. However, our formulation is capable of discovering items that require multiple skills. It is an empirical question that we leave for future work to understand how well Dynamic Cognitive Tracing performs in this context.

We are also interested in comparing Dynamic Cognitive Tracing to other automatic methods that produce cognitive models from data, such as matrix factorization techniques [27]. An interesting alternative we leave unexplored is finding a cognitive model by first clustering items into skills, and then using Knowledge Tracing with the discovered cognitive model. However, it is not clear how to learn the skill clustering from data that comes at different points of time. For example, it is not obvious how PCA could be applied to temporal data. To our knowledge, we are the first ones to propose a fully-unsupervised method that combines student modeling with discovering a cognitive model.

# 6. ACKNOWLEDGEMENTS

This work was supported in part by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080628 to Carnegie Mellon University. José was partially supported by the Costa Rican Ministry of Science and Technology (MICIT). The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or U.S. Department of Education. We thank the educators, students, and LISTENers who helped generate, collect, and analyze our data, and the reviewers for their helpful comments.

# 7. REFERENCES

[1] R. Baker, A. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In B. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, editors, *Intelligent Tutoring Systems*, volume 5091 of *Lecture Notes in Computer Science*, pages 406–415. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-69130-3. URL `http://dx.doi.org/10.1007/978-3-540-69132-7_44`.

[2] T. Barnes. The q-matrix method: Mining student response data for knowledge. In J. Beck, editor, *Proceedings of AAAI 2005: Educational Data Mining Workshop*, pages 978–980, Pittsburgh, PA, 2005.

[3] T. Barnes, D. Bitzer, and M. Vouk. Experimental analysis of the q-matrix method in knowledge discovery. In M.-S. Hacid, N. Murray, Z. Ras, and S. Tsumoto, editors, *Foundations of Intelligent Systems*, volume 3488 of *Lecture Notes in Computer Science*, pages 11–41. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-25878-0. URL `http://dx.doi.org/10.1007/11425274_62`.

[4] J. Beck. Difficulties in inferring student knowledge from observations (and why you should care). In *Educational Data Mining: Supplementary Proceedings of the 13th International Conference of Artificial Intelligence in Education*, pages 21–30, Marina del Rey, CA, 2007.

[5] J. Beck and K.-m. Chang. Identifiability: A fundamental problem of student modeling. In C. Conati, K. McCoy, and G. Paliouras, editors, *User Modeling 2007*, volume 4511 of *Lecture Notes in Computer Science*, pages 137–146. Springer Berlin / Heidelberg, 2007. URL `http://dx.doi.org/10.1007/978-3-540-73078-1_17`.

[6] J. Beck and J. Mostow. How who should practice: Using learning decomposition to evaluate the efficacy of different types of practice for different types of students. In B. Woolf, E. Ameur, R. Nkambou, and S. Lajoie, editors, *Intelligent Tutoring Systems*, volume 5091 of *Lecture Notes in Computer Science*, pages 353–362. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-69130-3.

[7] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis: A general method for cognitive model evaluation and improvement. In M. Ikeda, K. Ashley, and

T.-W. Chan, editors, *Intelligent Tutoring Systems*, volume 4053 of *Lecture Notes in Computer Science*, pages 164–175. Springer Berlin / Heidelberg, 2006. URL http://dx.doi.org/10.1007/11774303_17.

[8] A. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In M. Bauer, P. Gmytrasiewicz, and J. Vassileva, editors, *User Modeling 2001*, volume 2109 of *Lecture Notes in Computer Science*, pages 137–147. Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-42325-6. URL http://dx.doi.org/10.1007/3-540-44566-8_14.

[9] A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[11] M. Desmarais. Conditions for effectively deriving a q-matrix from data with non-negative matrix factorization. best paper award. In *EDM*, pages 41–50, Eindhoven, Netherlands, 2011.

[12] Z. Ghahramani and M. Jordan. Factorial hidden markov models. *Machine learning*, 29(2):245–273, 1997.

[13] S. Goldwater and T. Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1094.

[14] Y. Gong, J. Beck, and N. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In V. Aleven, J. Kay, and J. Mostow, editors, *Intelligent Tutoring Systems*, volume 6094 of *Lecture Notes in Computer Science*, pages 35–44. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-13387-9.

[15] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in recursive graphical models by local computations. *Computational Statistical Quarterly*, (4): 269–282, 1990.

[16] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2): 181–214, 1994.

[17] K. Koedinger, R. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. *A data repository for the EDM community: The PSLC DataShop*. CRC Press, Boca Raton, FL, 2010.

[18] K. Murphy. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

[19] P. Pavlik, H. Cen, and K. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. In *Proceeding of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*, pages 531–538. IOS Press, 2009.

[20] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 0-934613-73-7.

[21] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[22] J. Reye. Student modelling based on belief networks. *Int. J. Artif. Intell. Ed.*, 14:63–96, January 2004. ISSN 1560-4292. URL http://dl.acm.org/citation.cfm?id=1434852.1434856.

[23] A. Schein, L. Saul, and L. Ungar. A generalized linear model for principal component analysis of binary data. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 14–21, Key West, FL, 2003.

[24] A. P. Singh and G. J. Gordon. A unified view of matrix factorization models. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ECML PKDD '08, pages 358–373, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87480-5. URL http://dx.doi.org/10.1007/978-3-540-87481-2_24.

[25] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme. Recommender system for predicting student performance. *Procedia Computer Science*, 1 (2):2811 – 2819, 2010. ISSN 1877-0509. doi: 10.1016/j.procs.2010.08.006. URL http://www.sciencedirect.com/science/article/pii/S1877050910003194.

[26] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[27] T. Winters, C. Shelton, T. Payne, and G. Mei. Topic extraction from item-level grades. In J. Beck, editor, *American Association for Artificial Intelligence 2005 Workshop on Educational Datamining*, Pittsburgh, PA, 2005.

[28] Y. Xu and J. Mostow. Using logistic regression to trace multiple subskills in a dynamic bayes net. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. Stamper, editors, *Proceedings of the 4th International Conference on Educational Data Mining*, pages 241–245, Eindhoven, Netherlands, 2011.

# Identifying Learning Behaviors by Contextualizing Differential Sequence Mining with Action Features and Performance Evolution

John S. Kinnebrew
Department of EECS and ISIS
Vanderbilt University
1025 16th Ave S, Ste 102
Nashville, TN 37212
john.s.kinnebrew@vanderbilt.edu

Gautam Biswas
Department of EECS and ISIS
Vanderbilt University
1025 16th Ave S, Ste 102
Nashville, TN 37212
gautam.biswas@vanderbilt.edu

## ABSTRACT

Our learning-by-teaching environment, Betty's Brain, captures a wealth of data on students' learning interactions as they teach a virtual agent. This paper extends an exploratory data mining methodology for assessing and comparing students' learning behaviors from these interaction traces. The core algorithm employs sequence mining techniques to identify differentially frequent patterns between two predefined groups. We extend this technique by contextualizing the sequence mining with information on the student's task performance and learning activities. Specifically, we study transformation of action sequences using action features, such as activity categorizations, relevance and timing between actions, and repetition of analogous actions. We employ a piecewise linear segmentation algorithm in concert with the action transformation and differential sequence mining techniques to identify and compare segments of students' productive and unproductive learning behaviors. We present the results of this methodology applied to a recent middle school class study, in which students learned about climate change. Our primary focus in this analysis is the effectiveness and variation in the reading behaviors of high- versus low-performing students. These results illustrate the potential of this iterative methodology in identifying and interpreting learning behavior patterns at multiple levels of detail.

## 1. INTRODUCTION

Cognitive scientists have established that metacognition and self-regulation are important components for developing effective learning in the classroom and beyond [5; 18]. In developing a computer-based learning environment (CBLE) called Betty's Brain, we have adopted a self-regulated learning (SRL) framework to help students develop learning strategies. As they explore hypermedia resources on a science topic, they construct a causal map to teach Betty, their virtual Teachable Agent (TA) [4]. Betty only knows what she has been taught by the student, but, once taught, she can use this information to answer questions like "if *deforestation* increases, what effect does it have on *polar sea ice*?" and explain her answers as a chain of causal relations [9]. The student can also ask their TA to take quizzes, which are a set of questions created and graded by a Mentor Agent named Mr. Davis. The TA's quiz performance helps the students to assess and reflect on their TA's, and, therefore, their own learning performance. This assessment and subsequent reflection can help guide them as they continue their learning and teaching tasks. Previous studies have shown that observing Betty's quiz performance (which is actually a reflection of their own understanding) motivates students to learn more in order to help Betty improve her quiz score [4]. Overall, the combined learning and teaching task is complex, open-ended, and choice-rich, so learners must employ a number of cognitive and metacognitive skills to achieve success. At the cognitive level, they need to identify and understand relevant information from the resources in the system, represent that information in the causal map format to teach their agent, and use questions and quizzes to explore Betty's understanding and assess her overall progress. At the metacognitive level, they need to set goals and choose strategies related to their knowledge construction and monitoring tasks. In other words, they must decide when and how to acquire information, build and modify the causal map, check Betty's progress, and reflect on their own understanding of both the science knowledge and the evolving causal map structure. Their cognitive and metacognitive activities are scaffolded through dialogue and feedback provided by Mr. Davis. This feedback aims to help students progress in their learning, teaching, and monitoring tasks.

Betty's Brain is designed to track many details of students' learning interactions along with their teaching performance. This wealth of data provides opportunities to assess, model, and understand student learning behaviors and strategies more accurately. Realizing these opportunities requires effective methods for identifying interesting learning behavior patterns in the activity trace data. For example, sequential pattern mining [2] can be employed to identify frequent patterns in students' activity trace data. However, this can also result in a very large number of patterns[1] To overcome this

[1]Sequential pattern mining with activity traces of 16 $8^{th}$-grade students working in Betty's Brain identified over 1,000 patterns that occurred in at least 80% of the traces, when allowing gaps of one action to account for noise introduced by random or inconsequential actions.

problem, we have developed an algorithm that employs a novel combination of sequence mining techniques to identify differentially frequent patterns between groups of students (*e.g.*, experimental versus control conditions or high- versus low-performers) [8]. Further, this technique can be contextualized with information about the student's performance (*e.g.*, productive and counter-productive phases) over the course of their learning interactions [8].

In this paper, we extend these techniques by incorporating them in an iterative, exploratory methodology and further contextualizing the differential sequence mining with action features, such as activity categorizations, relevance and timing between actions, and repetition of analogous actions. We apply this exploratory data mining methodology to learning trace data gathered during a recent Betty's Brain study run in a middle school classroom. Previous analyses have shown that reading the resources occupies a significant portion of the students' learning activities. Therefore, we delve deeper than previous analyses by exploring reading action features (*e.g.*, short versus long reads and first reads versus rereads of a page) and analyze student behaviors and performance using this more detailed characterization of reading actions.

## 2. RELATED WORK

In this section, we briefly review relevant past work on using sequence mining techniques to analyze students' learning behaviors. For example, Perera *et al.* [14] investigate trace data from mirroring and feedback tools that support effective teamwork among students collaborating on software development using an open source professional development environment called TRAC. In their approach, they help all groups improve their work by observing and emulating the behaviors of the strong groups. They use k-means clustering to find groups of similar teams and similar individuals, and then employ a modified version of the Generalized Sequential Pattern (GSP) mining algorithm [16] to show that leadership and group interaction are important to success. Martinez *et al.* [12] discovered frequent sequences of actions that differentiate high-achieving groups from low-achieving groups of learners, who collaborate around a shared tabletop to answer an open question posed as a mystery problem. They apply a clustering algorithm to group similar patterns to aid in analyzing the pattern distribution across the groups. Employing sequential pattern mining allows them to identify differences between the higher- and lower-achieving groups in their manner of information gathering to solve the problem. Like Perera *et al.* [14] and Martinez *et al.* [12], we compare sequential patterns derived from groups of student activity sequences. However, our differential sequence mining algorithm directly incorporates comparisons between groups with additional metrics to identify interesting patterns, rather than manually performing researcher-directed comparisons after data mining.

Other researchers have employed sequential pattern mining (with a single set of student activity sequences or subsequences) to understand student learning behaviors. For example, Su *et al.* [17] propose a method for creating personalized activity trees to be used in a Sharable Content Object Reference Model (SCORM) e-learning system. They use sequential pattern mining to extract frequent learning patterns as part of a larger process that creates a decision tree to predict the group/category for a new student. Nesbit *et al.* [13] employ sequential pattern mining to investigate self-regulation in gStudy, which is a software application with similarities to Betty's Brain. In this system, students learn from multimedia documents and organize their knowledge with notes, concept maps, and other objects. Using sequential pattern mining, the authors hope to step beyond the question of whether a tool helps learners construct knowledge and instead investigate when and how learners use the tool as they self-regulate their knowledge construction activities. Similarly, our work investigates learning behaviors and self-regulation by identifying sequential patterns of student activity. However, unlike all of the preceding applications of sequential pattern mining, our methodology also analyzes students' evolving performance to identify, and group, action subsequences corresponding to productive and counter-productive phases. Further, our methodology iteratively employs action abstraction/transformation using features, such as activity categorizations, relevance and timing between actions, and repetition of analogous actions.

## 3. DIFFERENTIAL SEQUENCE MINING METHODOLOGY

To effectively perform sequential data mining on learning interaction traces, raw logs must first be transformed into an appropriate sequence of actions. Since these logs can contain a significant quantity of information about each student interaction with the system, as well as other system bookkeeping information, raising the level of abstraction from raw log events to a canonical set of distinct actions is a vital first step in effective analysis. Our methodology incorporates iterative refinement of this action abstraction step to focus the analysis on various learning activities and actions.

### 3.1 Action Abstraction with Context Summarization

Action abstraction is the first step of our data mining methodology, in which researcher-identified categories of actions define an initial alphabet (set of action symbols) for the sequences. This step filters out irrelevant information (*e.g.*, cursor position) and combines qualitatively similar actions (*e.g.*, querying an agent through different interfaces or about different concepts in a given topic).

To apply the abstraction process, log events captured by the CBLE are mapped to a sequence of canonical actions taken by each student. As in previous work, we abstract student activities in five primary categories [8]:

- *READ*: students access a page in the resources;
- *LINK or CONCept Edit*: students edit the causal map, with actions further divided by: (i) whether they operate on a causal link ("LINK") or concept ("CONC") and whether the action was an addition ("ADD"), removal ("REM"), or modification ("CHG"), *e.g.*, *LINKREM* or *CONCADD*;
- *QUER*: students use a template to ask Betty a question, and she uses a causal reasoning method to answer the question [9];
- *EXPL*: students probe Betty's reasoning by asking her to explain her answer to a question, and she uses dialogue and animation on the causal map to demonstrate her use of causal reasoning to answer the question;

- *QUIZ*: students assess how well they have taught Betty by having her take a quiz, which is a set of questions chosen and graded by the Mentor agent.

However, abstracting the raw log traces through action categorization, also strips potentially important context associated with the actions in the traces. For example, with the LINK-ADD action, the particular link added can provide important context information, such as whether this link relates to resource material the student read in a previous action. However, if the details of the exact link added are used to differentiate each edit action, we would end up with an unwieldy number of distinct actions, making it hard to discover and interpret behavior pattern sequences. To maintain a balance between the number of distinct actions and retaining relevant context information, we employ metrics that summarize context in order to distinguish actions. For example, we employ a relevance summarization metric, which establishes whether the content/object of an action is related to a small number of recent activities, where recent is defined by a configurable window of previous actions [3]. This relevance metric splits each categorized action into two distinct actions: (1) relevant to at least one of the recent actions (with the "-REL" suffix) and (2) irrelevant to any of the recent actions (with the "-IRR" suffix).

In this methodology, the choice of specific context-summary metrics and their application to different categories of action is iteratively refined over repeated analyses of the interaction traces. This allows the researcher to focus the analysis, providing more detail and context associated with specific learning activities or strategies. In previous work, we presented an initial analysis of student action sequences applying only the relevance metric, which illustrated some interesting map editing and monitoring behaviors distinguishing high-performing and low-performing students [8]. However, that analysis did not differentiate between reading actions (*e.g.*, long versus short, or reading pages in sequence versus using keyword search), which are frequent and vital to student learning in Betty's Brain [8; 15].

In this paper, we present the results of a subsequent iteration in this extended methodology, in which we apply additional metrics to distinguish different types of READ actions. As a continuation of the exploratory methodology, a future iteration might instead focus on actions related to editing the causal map by applying additional editing metrics (*e.g.*, whether the edit increased or decreased the correspondence between the student's map and the expert map, or whether the edit introduced a cycle, continued a chain of causal relationships, or added a branch to a chain of causal relationships). However, to maintain a reasonable number of distinct actions in that hypothetical iteration (such that sufficiently frequent patterns could still be identified), the number of reading-related metrics would be correspondingly reduced.

In the analysis and iteration of the methodology presented in this paper, we apply three reading-related metrics to the student action sequences:

- *Source (TOC/HLNK/HIST)*: how the student reached the page he/she is reading - by selecting a page in the table of contents (TOC) always displayed on the left of the resources, from a hyperlink (HLNK) on another page, or using the backward or forward button to move through their history of pages (HIST) like a web browser;

- *Time (SHRT/FULL)*: a determination of whether the student spent enough time on the page to have read a significant amount of the material[2] (FULL) or only spent a brief period of time on the page (SHRT), possibly skimming the material or checking whether the page was one for which they were searching;

- *Repetition (FRST/REPT)*: a determination of whether the student had never done a FULL read of the page (FRST) or this was a reread of the page (REPT) because the student had previously done a FULL read;

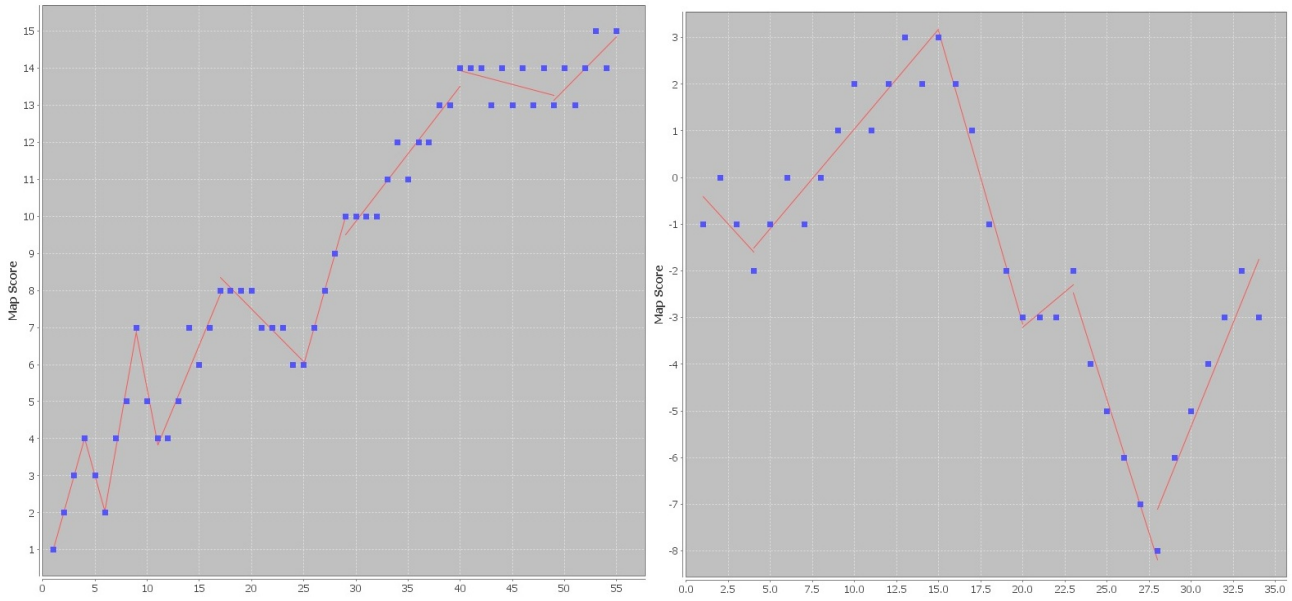
In addition to metrics related to individual actions, we also apply another, general transformation to the action sequence. In an environment like Betty's Brain, there are cases in which students often perform a particular type of action (*e.g.*, adding concepts) repeatedly in sequence, which can result in a variety of frequent patterns that differ only by the number of repetitions of that action. To improve this exploratory analysis, our action abstraction step distinguishes a single action from repeated actions, which are condensed to a single action with the "-MULT" suffix. Using the re-transformed sequences, our differential sequence mining technique can more efficiently identify trends that could otherwise be hidden by the multitude of frequent patterns differing only in the length of a repeated action sequence [8].

## 3.2 Differential Sequence Mining

To identify important activity patterns in a comparison between two sets of action sequences, our methodology employs a novel combination of sequence mining techniques. Sequential pattern mining [2] methods find the most frequent action patterns across a set of action sequences, while episode mining [11] discovers the most frequently used action patterns within a given sequence. However, finding the patterns most important for interpreting learning behaviors or differentiating between groups of students is challenging, because of the need to limit the large set of frequent patterns to ones that are interesting and important (*i.e.*, our focus is on the *effectiveness* of mining techniques in identifying these important patterns, rather than the *efficiency*, or speed, in calculating the frequent patterns [1]).

In comparing across groups of action sequences, such as high- versus low-performing students, the differences between the groups provide a natural criterion for identifying important patterns that may elucidate differences in learning behavior. To use this criterion for mining important frequent patterns, we define two measures of frequency and the corresponding differences calculated across the groups. The sequential pattern mining frequency measure (*i.e.*, the number of sequences in which the pattern occurs, regardless of how many times) is important for identifying patterns common to a group of action sequences. We refer to this as the "sequence support" (*s-support*) of the pattern, following the convention of [10], and we call patterns meeting a given s-support threshold *s-frequent*. The second metric is the

[2]Based on the length of typical resource pages and the reading abilities of the students in the study, we set the threshold between short and full reads to be 30 seconds. Further, the large majority of reads in the short category were actually under 5 seconds and most of the reads in the full category were over a minute.




(a) High-performing student (b) Lo-performing student

1: Example student performance evolution with identified phases

episode frequency, defined as the number of times the pattern is repeated *within* an action sequence. We refer to this frequency measure as the "instance support" (*i-support*), following [10]. To calculate the i-support of a pattern in a *group* of traces, we use the mean of the pattern's i-support values across all sequences in the group.

The details of our differential sequence mining algorithm are presented in [8], but we briefly outline the main steps of the algorithm. First, a sequential pattern mining algorithm (SPAMc [6]) identifies the patterns that meet a minimum s-support constraint within each group, employing a maximum gap constraint to account for noise, which is interpreted as a small number of irrelevant actions that may be interspersed in a pattern. In this paper, we employ a gap constraint of 1, *i.e.*, we allow at most one irrelevant action between each consecutive action in a pattern. To compare the identified s-frequent patterns across groups, we calculate the mean i-support of every pattern for each group. In order to identify patterns whose usage more clearly differ between the two groups, we also filter the patterns based on the p value of a t-test comparing pattern i-support between the groups.

This comparison produces four distinct categories of frequent patterns: two categories where the patterns are s-frequent in only one group, illustrating patterns primarily employed by that group, and two categories where the patterns are common to both groups but used more often in one group than the other. The patterns in each of these qualitatively distinct categories are (separately) sorted by the difference in mean group i-support to focus the analysis on the most differentially frequent patterns.

## 3.3 Performance Evolution Phases

In the Betty's Brain environment, a student's work can be assessed in terms of their performance on the learning task, which we define as the student's current map score[3]. By tracking the evolution of students' map scores, we can quantify how their learning and map-building performance develop as they work on the system. To more effectively identify and contextualize learning behavior patterns, we consider phases of productive (increasing map score) and counter-productive (decreasing map score) activity over the course of learning by tracking their map scores, as illustrated in Figure 1.

These phases are identified by generating a piecewise, linear representation (PLR) for a sequence of two-dimensional points. In this representation, the x-value is a cumulative measure of student editing activity (*i.e.*, the number of edit actions the student has performed thus far) and the y-value is the student's total map score after the corresponding edit action [8]. Figures 1(a) and 1(b) illustrate these performance phases with plots of map score versus number of edits for a high-performing and a low-performing student, respectively. To generate this representation, we employ a standard bottom-up, time-series linear segmentation algorithm [7] with the sum-squared-error (SSE) of the segments as the criterion metric [8].

## 3.4 Summary of Methodology

Our iterative methodology consists of four major steps to identify learning behaviors contextualized by performance evolution between groups of students:

1. *Action abstraction*: Logfiles are processed to produce a sequence of actions for each student by mapping sets of interaction events to canonical actions. Each canonical action is contextualized and split into distinct actions by applying metrics, such as the relevance

[3]The map score is defined as the number of correct links (based on the expert map) in the student's map minus the number of incorrect links.

metric and the reading metrics. At each iteration additional metrics can be applied, as well as previous metrics removed, based on the results of previous iterations. Finally, any subsequences of a repeated action are condensed into a single "action" identified with the "-MULT" suffix.

2. *Performance phase identification*: Student action sequences are split into subsequences using the time-series segmentation algorithm. These subsequences are filtered to produce two sequential datasets: a) productive action sequences corresponding to segments with a positive progress slope above a given cutoff, and b) counter-productive action sequences corresponding to segments with a negative progress slope below a given (negative) cutoff.

3. *Differential sequence mining*: The student groups, as well as productive and counter-productive action subsequences within those groups, are compared to identify differentially frequent patterns of action.

4. *Interpretation*: The differentially frequent sequential patterns of action are interpreted in terms of effective and ineffective learning behaviors exhibited by students during the learning task. Investigation of pattern details (i.e., raw event details for instances of these patterns) may yield further insights into student cognition and metacognition, as well as potential flags and triggers for adaptive feedback/scaffolding in the system.

# 4. RESULTS

We illustrate our methodology using interaction trace data from a recent study with 40 $8^{th}$-grade students taught by the same teacher in a middle Tennessee school. At the beginning of the study, students were introduced to the science topic (global climate change) during regular classroom instruction, provided an overview of causal relations and concept maps, and given hands-on training with the system. For the next five days, students taught their agent about climate change and received feedback on metacognitive strategies from the Mentor agent. In this version of the system, the majority of the metacognitive feedback was related to knowledge construction strategies [15]. However, the Mentor agent also provided advice on monitoring strategies to help students recognize and correct errors in their casual maps.

The results of this study presented an interesting dichotomy in student performance at constructing their causal concept maps. 16 of the students taught their agent a correct, complete map or one very close to it (these students achieved map scores between 11 and 15, inclusive, where 15 was the maximum possible score). Another 18 students taught their agents relatively poor maps with a map score of 5 or below. Only 6 students had a map score in between these groups (*i.e.*, a map score of 6 to 10, inclusive). Therefore, we focus on an analysis and comparison of the learning activities of the high-performing ("Hi") student group and the low-performing ("Lo") student group. An initial analysis of the activity traces from this study was presented in [8]. Here we focus on the effectiveness and variation in students' reading behaviors by refining the action abstraction step in our exploratory methodology with additional (reading-related) metrics, discussed in Section 3.1. We should note that students in the "Hi" group had higher pre-test scores in all of the categories as compared to the "Lo" group. However, a detailed analysis shows that 40% of the links added by the "Hi" group were initially incorrect (this number was 58% for the "Lo" group). This shows that the "Hi" group had to put in significant effort into discovering errors in their maps and correcting them, and the final results show that they were quite successful in their monitoring and correction tasks. This was not the case for the "Lo" group. Therefore, a comparison of the learning behaviors of the two group should demonstrate an important dichotomy in the strategies employed by the two groups that mirrors the dichotomy in their performance. To further differentiate behaviors associated with high and low performance, we compared productive and counter-productive phases of student activities. We discuss the results of our analyses in greater detail below.

To assess students' overall learning gains, calculated as normalized gains[4] in pre- to post-test scores, we categorize the pre- and post-test questions into three groups: (i) definition questions about the science topic in multiple choice (MC) format, (ii) questions requiring reasoning about the science topic that students had to answer by writing sentences ("short answer"), and (iii) questions about causal reasoning using a causal map that was not related to the science topic. Table 1 presents the students average scores (and standard deviations). The results of an ANOVA comparing the Hi and Lo student groups on each of the pre-post gains show significant differences between the Hi and Lo groups only for the definitional MC questions. Table 1 also presents ANOVA analyses of the difference in performance for the map-building metrics: (i) link accuracy - the percentage of links added to the map that were correct; (ii) link creation effort - the total number of student actions divided by the number of correct link edits, a measure of the effort by the student in order to produce a correct link edit; and (iii) action relevance - the percentage of student actions that were relevant (as described in Section 3.1) to at least one of the three previous actions show significant differences in favor of the Hi group with moderate effect sizes. These results indicate that students in the Hi group were more accurate in their map edits and generally more efficient in their learning and teaching activities. Further, they tended to employ a more systematic approach to the task, as indicated by their higher action relevance score. Overall, students who achieved success in teaching Betty accurate causal maps also learned significantly more factual information, but their gains in causal reasoning and short answer questions were not significantly different from the low-performing group [8; 15].

As a first analysis to elucidate broad differences in reading behavior between the Hi and Lo groups, Table 2 presents the relative proportion of reading activities categorized by each metric presented in Section 3.1. Both groups performed roughly equal numbers of read actions on pages they had previously read in-depth ("Repeat" (REPT)) compared to ones they had not read in-depth ("First" (FRST)). The Lo group relied slightly more on short (SHRT) reads (74%) than the Hi group (69%), and the ratio of short to full page (FULL) reads was approximately 3:1 for the Lo group and 2:1 for the Hi group. Similarly, the Lo group's read actions

[4]The normalized learning gain was calculated as: $\frac{(post-pre)}{(max-pre)}$

1: High vs. Low Performers - Learning Gain and Map Score

| Metric | Hi Group | Lo Group | F | Sig. | Effect Size (Cohen's f) |
|---|---|---|---|---|---|
| Definition MC Norm. Gain | 0.535 (0.344) | -0.202 (0.769) | 12.448 | 0.001 | 0.624 |
| Causal Reasoning Norm. Gain | 0.130 (0.614) | -0.029 (0.414) | 0.799 | 0.378 | 0.157 |
| Short Answer Norm. Gain | 0.027 (0.241) | -0.028 (0.134) | 0.700 | 0.409 | 0.146 |
| Map Score | 14.500 (1.155) | 2.780 (1.592) | 590.171 | 0.001 | 4.314 |
| Link Accuracy (LA) | 60.3% (7.1%) | 42.8% (10.6%) | 31.528 | 0.001 | 0.992 |
| Link Creation Effort (LCE) | 11.630 (8.196) | 20.665 (5.942) | 12.745 | 0.001 | 0.652 |
| Action Relevance | 53.3% (8.4%) | 40.6% (12.2%) | 12.401 | 0.001 | 0.622 |

were deemed more irrelevant (IRR) to recent actions (again the ratio of IRR to REL (relevant) reads was 3:1). The same ratio for the Hi group was 2:1.

2: Relative Proportion of Actions by Reading Metrics

| Group | First : Repeat (FRST : REPT) | Short : Full (SHRT : FULL) | Irrel. : Rel. (IRR : REL ) |
|---|---|---|---|
| Hi | 51% : 49% (SD 12%) | 69% : 31% (SD 15%) | 67% : 33% (SD 9%) |
| Lo | 55% : 45% (SD 14%) | 74% : 26% (SD 12%) | 74% : 26% (SD 7%) |

To analyze specific reading behaviors illustrated by these students' interaction traces, we applied the differential sequence mining technique described in Section 3.2. This allowed us to identify a variety of interesting learning behaviors related to reading that were not apparent from the higher level analyses of behavior patterns we had conducted in the past [8; 15]. Table 3 presents the top five patterns in each of the differential categories detailed in Section 3.2. For the analysis, we employed an s-support threshold of 50% to analyze patterns that were evident in the majority of either group of students and employed a standard statistical significance cutoff of $p < 0.05$. In all of the differential sequence mining results presented here, we employed a maximum gap threshold of 1, to allow for "noise" from irrelevant or interchangeable actions in the learning activity sequences, as described in Section 3.2.

All reads in the differentially frequent patterns distinguishing reading behaviors between high and low students were pages selected from the table of contents (TOC) rather than from hyperlinks within pages or the (backward/forward) history mechanism. This is unsurprising since raw frequencies of these different types of reading activities indicated that in both the high and low group, the large majority of reading activities involved selecting pages from the table of contents. Table 3 shows that the high group was much more likely to add a link (both relevant (REL) or irrelevant (IRR) links with respect to recent actions) following a full-length (FULL) re-read (REPT) of a page that was relevant (REL) to recent actions. This greater reliance on extended re-reads before adding links suggests the high group employed a more careful approach to identifying causal links in the resources, which may have helped increase their accuracy in teaching correct links, and also their ability to correct previously taught incorrect links.

Further, the high group more frequently employed reading activities in a monitoring context (*i.e.*, in conjunction with quiz actions). Besides following extended re-reads by adding links, the high group was also more likely to follow them with quizzes, possibly in an attempt to connect what they were reading with their TA's right and wrong answers based on the current map. Following quizzes, they were more likely to do a quick re-read of a relevant page, which suggests another monitoring strategy, such as confirming links used by the TA in quiz answers. The differentially frequent patterns employed more by the low group were various combinations of reading, especially short reads and ones not relevant to recent actions. This may be indicative of a less consistent approach to reading and of strategies that do not systematically combine reading with other knowledge construction and monitoring activities.

To further investigate which reading behaviors may have contributed to the high performers' success, we identified differentially frequent patterns when students were productive as opposed to being counter-productive during their map building activities. The method for extracting the productive versus counter-productive phases was described in Section 3.3, and we included all segments with a slope greater than or equal to 0.4 in the productive set and all segments with a slope less than or equal to -0.4 in the counter-productive set[5] [8]. For the differential sequence mining with performance evolution subsequences analysis, we employed a lower s-support threshold because the sequences were significantly shorter than the complete student activity sequences. Specifically, we employed an s-support threshold of 20% to analyze patterns that occurred with some regularity (*i.e.*, in at least one out of every five subsequences). Similarly, given the limited length and number of sequences, we employed a relaxed cutoff on the t-test comparison of $p < 0.10$.

In comparing the Hi group's productive to counter-productive periods, the only differentially frequent pattern observed was that extended, relevant rereads (READ-TOC-REPT-FULL-REL) occurred approximately twice as frequently ($p = 0.034$) in productive segments ($i - support = 0.65$) than counter-productive segments ($i - support = 0.38$). This reliance on extended, relevant re-reads, especially during productive periods, provides further evidence that a more careful, systematic approach to reading may have been particularly beneficial for the high-performing students. In comparing the Lo group's productive to counterproductive periods, the only differentially frequent pattern observed was that extended, relevant reading of a page for the first time occurred approximately five times as frequently ($p = 0.039$) in productive segments ($i - support = 0.28$) than counter-productive segments ($i - support = 0.06$). This suggests that when the low-performing students read a page in-depth for the first time

[5]The slope cutoff of 0.4/-0.4 was determined by qualitative analysis of a sample of student map score plots to distinguish generally productive/counter-productive segments [8]

3: High vs. Low Performers - Differentially Frequent Patterns

| Pattern | I-Support (Hi) | I-Support (Lo) | t-test (p value) | S-Freq Group |
|---|---|---|---|---|
| READ-TOC-REPT-FULL-REL → LINKADD-REL | 2.19 | 0.61 | 0.005 | Hi |
| READ-TOC-REPT-FULL-REL → QUIZ | 1.50 | 0.39 | 0.010 | Hi |
| QUIZ → READ-TOC-REPT-SHRT-REL | 1.44 | 0.33 | 0.018 | Hi |
| READ-TOC-REPT-SHRT-REL → LINKADD-IRR | 1.13 | 0.17 | 0.008 | Hi |
| READ-TOC-REPT-FULL-IRR → LINKREM-IRR | 1.13 | 0.28 | 0.016 | Hi |
| READ-TOC-REPT-FULL-REL → LINKADD-IRR | 2.00 | 0.94 | 0.022 | BOTH |
| READ-TOC-FRST-SHRT-IRR → READ-TOC-REPT-SHRT-IRR-MULT | 0.31 | 0.89 | 0.043 | Lo |
| READ-TOC-REPT-SHRT-IRR → READ-TOC-REPT-SHRT-IRR-MULT | 0.31 | 1.00 | 0.040 | Lo |
| READ-TOC-REPT-FULL-IRR → READ-TOC-FRST-SHRT-IRR | 0.44 | 1.17 | 0.033 | Lo |
| READ-TOC-REPT-FULL-IRR → READ-TOC-REPT-SHRT-IRR | 0.63 | 1.67 | 0.013 | Lo |
| READ-TOC-REPT-SHRT-REL → READ-TOC-FRST-SHRT-REL | 0.50 | 1.78 | 0.047 | Lo |

they tended to extract useful information, especially if the page was relevant to their recent activities. However, they did not seem to be as effective as the high-performing students in extracting additional information when re-reading pages.

To gain further insight into differences in productive reading behaviors exhibited by students in Betty's Brain, we applied the differential sequence mining technique to compare the productive subsequences between the Hi and Lo groups. This analysis, presented in Table 4, illustrates four reading activities more frequently employed by the low-performing students during their productive activity periods. The greater frequency of irrelevant, extended rereads (both from the table of contents and from hyperlinks) in the Lo group, suggests that although they tended to be less systematic in their approach, they were still able to gain some useful information from rereading the resources. However, their greater reliance on initial, in-depth reads (READ-TOC-FRST-FULL-REL) in productive periods suggests they may have had a harder time identifying the less obvious causal relations often found after multiple reads. Finally, the repeated use of short, irrelevant reads during productive periods by the Lo group may indicate an inefficient, but ultimately effective, searching behavior.

## 5. CONCLUSIONS

In this paper, we extended an exploratory data mining methodology for identifying important learning behaviors with an iterative approach to action abstraction using a variety of action features and presented results analyzing reading behaviors of students in a learning-by-teaching environment. The exploratory methodology combines iterative action abstraction, a sequence mining technique to identify differentially frequent activity patterns, and piecewise linear segmentation of activity phases with respect to the evolution of a performance or progress measure. Results from a recent classroom study with Betty's Brain illustrate the effectiveness of this methodology and iterative action abstraction for identification of important learning behaviors at multiple levels of analysis, including behaviors distinguished by their relationship to productive or counter-productive phases of activity.

Although the majority of frequent action patterns that could be identified in these interaction traces are common to both high-performing students and low-performing students (and occur throughout the course of students' interaction with the system), the analysis employing our differential sequence mining methodology elucidated some important learning behaviors. In this paper, we extended previous results with a focus on reading behaviors related to successful performance in the learning environment and differentiating them from ones employed by less successful students. Overall, high-performing students differentially employed reading behaviors that indicated a more careful and systematic strategy of reading. Their activity patterns more frequently involved re-reading pages from the resources, such as employing full-length re-reads of the resources before adding a link. Further, the reading activity patterns distinguishing high-performers from low-performers usually involved reading pages that were relevant to recent actions, suggesting a more systematic reading behavior overall. Productive periods were particularly distinguished in high performers by a larger number of full-length, relevant re-read actions.

Performance was also linked to monitoring behaviors that incorporated re-reading of the resource material. In particular, high-performers were more likely to employ various types of reading actions both before and after assessments of progress/correctness using the quiz. Low performers, on the other hand, had a differential tendency to use irrelevant, extended re-reads of pages in the resources during productive periods. They may have also had more difficulties identifying the less obvious causal relations, as suggested by their greater reliance on initial reads of a resource page during productive periods.

In future work, we also intend to expand upon the presented data mining techniques through a variety of enhancements and additional applications. We will enhance the existing summarization of action-relevance to include determination of the relationship between the specific actions in all subsequences matching the identified patterns. For example, this enhanced action-relevance summarization will allow us to determine how frequently a pattern like READ-TOC-REPT-FULL-REL → LINKADD-REL involves adding a correct link and whether the specific link added was discussed in the resource page from the reading action. Relating identified patterns of action back to specific details and context in the interaction traces could provide significant benefits for more efficient and effective interpretation of learning behaviors. Based on this analysis, we will expand and revise the feedback triggering conditions and student modeling to improve learning behavior feedback from the Betty's Brain agents.

4: High vs. Low Performers (Productive Segments) - Differentially Frequent Patterns

| Pattern | I-Support (Hi) | I-Support (Lo) | t-test (p value) | S-Frequent Group |
|---|---|---|---|---|
| READ-TOC-REPT-FULL-IRR | 0.51 | 1.00 | 0.070 | BOTH |
| READ-TOC-FRST-FULL-REL | 0.10 | 0.28 | 0.091 | Lo |
| READ-HLNK-REPT-FULL-IRR | 0.08 | 0.28 | 0.057 | Lo |
| READ-TOC-REPT-SHRT-IRR-MULT | 0.15 | 0.39 | 0.079 | Lo |

## 6. ACKNOWLEDGEMENTS

This work has been supported by NSF-IIS Award #0904387.

## 7. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of The ACM SIGMOD International Conference on Management of Data*, volume 22, pages 207–216. ACM, 1993.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh IEEE International Conference on Data Engineering (ICDE)*, pages 3–14, 1995.

[3] G. Biswas, H. Jeong, J. Kinnebrew, B. Sulcer, and R. Roscoe. Measuring Self-regulated Learning Skills through Social Interactions in a Teachable Agent Environment. *Research and Practice in Technology-Enhanced Learning (RPTEL)*, 5(2):123–152, 2010.

[4] G. Biswas, K. Leelawong, D. Schwartz, N. Vye, and T. Vanderbilt. Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence*, 19(3):363–392, 2005.

[5] J. Bransford, A. Brown, and R. Cocking, editors. *How people learn*. National Academy Press Washington, DC, Washington, D.C., 2000.

[6] J. Ho, L. Lukov, and S. Chawla. Sequential pattern mining with constraints on large protein databases. In *Proceedings of the 12th International Conference on Management of Data (COMAD)*, pages 89–100, 2005.

[7] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*, volume 57, pages 1–22. World Scientific, 2004.

[8] J. S. Kinnebrew, K. M. Loretz, and G. Biswas. A contextualized, differential sequence mining method to derive students' learning behavior patterns. *Journal of Educational Data Mining*, In Press, 2012.

[9] K. Leelawong and G. Biswas. Designing learning by teaching agents: The Betty's Brain system. *International Journal of Artificial Intelligence in Education*, 18(3):181–208, 2008.

[10] D. Lo, S. Khoo, and C. Liu. Efficient mining of recurrent rules from a sequence database. In *Proceedings of the 13th International Conference on Database Systems for Advanced Applications*, pages 67–83. Springer-Verlag, 2008.

[11] H. Mannila, H. Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[12] R. Martinez, K. Yacef, J. Kay, A. Al-Qaraghuli, and A. Kharrufa. Analysing frequent sequential patterns of collaborative learning activity around an interactive tabletop. In *Proceedings of the Fourth International Conference on Educational Data Mining*, Eindhoven, Netherlands, 2011.

[13] J. Nesbit, M. Zhou, Y. Xu, and P. Winne. Advancing log analysis of student interactions with cognitive tools. *12th Biennial Conference of the European Association for Research on Learning and Insruction (EARLI)*, 2007.

[14] D. Perera, J. Kay, I. Koprinska, K. Yacef, and O. Zaïane. Clustering and sequential pattern mining of online collaborative learning data. *IEEE Transactions on Knowledge and Data Engineering*, 21(6):759–772, 2009.

[15] J. R. Segedy, J. S. Kinnebrew, and G. Biswas. Supporting cognitive and metacognitive skills in complex, open-ended learning environments. *Journal of Educational Psychology*, In Review, 2012.

[16] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT): Advances in Database Technology*, pages 3–17. Springer-Verlag, 1996.

[17] J.-M. Su, S.-S. Tseng, W. Wang, J.-F. Weng, J. Yang, and W.-N. Tsai. Learning portfolio analysis and mining for scorm compliant environment. *Journal of Educational Technology and Society*, 9(1):262–275, 2006.

[18] B. Zimmerman. Theories of self-regulated learning and academic achievement: An overview and analysis. In B. Zimmerman and D. Schunk, editors, *Self-regulated learning and academic achievement: Theoretical perspectives*, pages 1–37. Erlbaum, Mahwah, NJ, 2001.

# Identifying Students' Characteristic Learning Behaviors in an Intelligent Tutoring System Fostering Self-Regulated Learning

François Bouchet, Roger Azevedo
Department of ECP
SMART Laboratory
McGill University
Montreal, H3A 1Y2, Canada
francois.bouchet@mcgill.ca

John S. Kinnebrew, Gautam Biswas
Department of EECS/ISIS
Vanderbilt University
Nashville, TN 37203, USA
john.s.kinnebrew@vanderbilt.edu

## ABSTRACT

Identification of student learning behaviors, especially those that characterize or distinguish students, can yield important insights for the design of adaptation and feedback mechanisms in Intelligent Tutoring Systems (ITS). In this paper, we analyze trace data to identify distinguishing patterns of behavior in a study of 51 college students learning about a complex science topic with an agent-based ITS that fosters self-regulated learning (SRL). Preliminary analysis with an Expectation-Maximization clustering algorithm revealed the existence of three distinct groups of students, distinguished by their test and quiz scores (low for the first group, medium for the second group, and high for the third group), their learning gains (low, medium, high), the frequency of their note-taking (rare, frequent, rare) and note-checking (rare, rare, frequent), the proportion of sub-goals attempted (low, low, high), and the time spent reading (high, high, low). In this paper, we extend this analysis to identify characteristic learning behaviors and strategies that distinguish these three groups of students. We employ a differential sequence mining technique to identify differentially frequent activity patterns between the student groups and interpret these patterns in terms of relevant learning behaviors. The results of this analysis reveal that high-performing students tend to be better at quickly identifying the relevance of a page to their subgoal, are more methodical in their exploration of the pedagogical content, rely on system prompts to take notes and summarize, and are more strategic in their preparation for the post-test (*e.g.*, using the end of their session to briefly review pages). These results provide a first step in identifying the group to which a student belongs during the learning session, thus making possible a real-time adaptation of the system.

## 1. INTRODUCTION

Use of metacognition and self-regulated processes has been identified as a key element for successful learning in general [**?**; **?**; **?**; **?**]. In the particular context of an intelligent tutoring system (ITS), it means it is crucial to ensure that students are actively using key self-regulated learning (SRL) processes, which can be achieved through prompts, scaffolding, and feedback. A major challenge is to make the ITS more adaptive to individual learning characteristics, such as browsing behavior and initiative in performing appropriate SRL processes.

Using MetaTutor, an agent-based ITS that fosters the use of SRL processes, we have collected a large amount of data from students interacting with the system while they were learning about the human circulatory system. In this paper, our goal is to answer two questions: (1) how can students be grouped according to their performance and their type of interaction with the system? and (2) how do specific learning behaviors of high- and low-performing students differ, in particular regarding their use of SRL processes in MetaTutor?

In this paper, we propose to answer the first question using a clustering approach that groups students with similar performance and scores on other system interaction metrics. For the second question, we analyze members of the three clusters (especially comparing high- and low-performing students) with a differential sequence mining method [**?**], which identifies statistically significant differences in frequent behaviors between clusters.

This paper is organized as follows. In section 2, we start by discussing related work that combines clustering and pattern mining techniques for analysis of data from computer-based learning environments. In section 3, we introduce the ITS used for data collection, MetaTutor, as well as theoretical grounding of its key features, which encourage learners to perform self-regulation monitoring and strategy as they learn with the system. Section 4 describes the data collected and the relevant events encoded as actions, as well as the clustering performed to distinguish different types of students. Section 5 presents the principles of the method of differential sequence mining, its application to the data, and the results obtained in terms of patterns of actions that distinguish students from different clusters. Section 6 then discusses the practical implications of those findings in terms of potential modifications to the ITS, before concluding in section 7.

## 2. RELATED WORK

Analysis of trace log data from users' interactions to better understand their learning process and distinguish groups of learners (*e.g.*, efficient versus inefficient ones) has been an

important area of research in educational data mining. For example, Perera *et al.* [**?**] follow a 2-step methodology like ours, as they start by using a clustering algorithm (k-means) to identify strong groups of students collaborating in a software development task using an open environment (TRAC). The students are first clustered according to a set of attributes extracted a posteriori, and then they use a modified version of the Generalized Sequential Pattern mining algorithm [**?**] to identify frequent sequences of actions that characterize the most successful groups. In [**?**], Romero *et al.* also use a combination of clustering and sequential pattern mining to identify different kinds of browsing behavior that students exhibit in their learning environment, "AHA!", in order to provide them links to the most appropriate pages. With gStudy, Nesbit *et al.* [**?**] are interested in the use of self-regulation by students learning from multimedia documents. They apply sequential pattern mining to find common subsequences between groups of students, although they do not perform any clustering beforehand. Martinez *et al.* [**?**] pursue a similar approach and objective, as they aim to discover frequent sequences of actions that distinguish a group of students with high achievements from one with low achievements. They use a combination of pattern mining and clustering techniques to identify the most successful strategies in the context of a collaborative learning tool on a tabletop device. However, they first extract frequent patterns of actions and then cluster them in order to examine clusters of patterns associated with each group. Tang and McCalla [**?**] also use sequence mining and then clustering in their web learning environment, to facilitate instructional planning and diagnose students behaviors.

# 3. METATUTOR ENVIRONMENT

## 3.1 General overview

MetaTutor is a multi-agent, adaptive hypermedia learning environment, which presents challenging human biology science content. The primary goal underlying this environment is to investigate how multi-agent system can adaptively scaffold SRL and metacognition within the context of learning about complex biological content. MetaTutor is grounded in a theory of SRL that views learning as an active, constructive process whereby learners set goals for their learning and then attempt to monitor, regulate, and control their cognitive and metacognitive processes in the service of those goals [**?**; **?**; **?**]. More specifically, MetaTutor is based on several theoretical assumptions of SRL that emphasize the role of cognitive, metacognitive (where metacognition is conceptualized as being subsumed under SRL), motivational, and affective processes [**?**; **?**]. Moreover, learners must regulate their cognitive and metacognitive processes in order to integrate multiple informational representations available from the system. While all students have the potential to regulate, few students do so effectively, possibly due to inefficient or a lack of cognitive or metacognitive strategies, knowledge, or control.

As a learning tool, MetaTutor has a multitude of features that embody and foster self-regulated learning (*cf.* Figure **??**). These include four pedagogical agents which guide students through the learning session and prompt students to engage in planning, monitoring, and strategic learning behaviors. In addition, the agents can provide feedback and engage in a tutorial dialogue in an attempt to scaffold students' selection of appropriate sub-goals, accuracy of metacognitive judgments, and use of particular learning strategies. The system also uses natural language processing to allow learners to express metacognitive monitoring and control processes. For example, learners can type that they do not understand a paragraph and can also use the interface to summarize a static illustration related to the circulatory system. Additionally, MetaTutor collects information from user interactions with it to provide adaptive feedback on the deployment of students' SRL behaviors. For example, students can be prompted to self-assess their understanding (*i.e.*, system-initiated judgment of learning [JOL]) and are then administered a brief quiz. Results from the self-assessment and quiz allow pedagogical agents to provide adaptive feedback according to the calibration between students' confidence of comprehension and their actual quiz performance.

During learning, MetaTutor is capable of measuring the deployment of self-regulatory processes by allowing us to collect rich, multi-stream data, including: self-report measures of SRL, on-line measures of cognitive and metacognitive processes (*e.g.*, concurrent think-alouds), dialogue moves regarding agent-student interactions, natural language processing of help-seeking behavior, physiological measures of motivation and emotions, emerging patterns of effective problem solving behaviors and strategies, facial data on both basic (*e.g.*, anger) and learning-centered emotions (*e.g.*, boredom), and eye-tracking data regarding the selection, organization, and integration of multiple representations of information (*e.g.*, text, diagrams). The collection of these various data streams is critical to enhancing our understanding of when, how, and why students regulate or do not regulate their learning and adapt their regulatory behaviors. These data are then used to develop computational models designed to detect, track, model, and foster students' SRL processes during learning.

## 3.2 Self-Regulated Learning with MetaTutor

This paper is theoretically-guided by contemporary models of SRL that emphasize the temporal deployment of these processes during learning [**?**]. As such, the goal is to use multiple measures to detect, track, and model learners' use of cognitive, affective, and metacognitive (CAM) processes during learning. As such, we use Winne and Hadwin's model [**?**; **?**] because it proposes that learning occurs in four basic phases: (1) task definition, (2) goal-setting and planning, (3) studying tactics, and (4) adaptations to metacognition. Their model emphasizes the role of metacognitive monitoring and control as the central aspects of learners' ability to learn complex material across different instructional contexts (*e.g.*, using a multi-agent system to track and foster SRL) in that information is processed and analyzed within each phase of the model. Recently, Azevedo and colleagues [**?**; **?**; **?**; **?**; **?**] extended this model and provided extensive evidence regarding the role and function of several dozen CAM processes during learning with student-centered learning environments (*e.g.*, multimedia, hypermedia, simulations, intelligent tutoring systems).

In brief, our model makes the following assumptions: (1) successful learning involves having learners monitor and control (regulate) key CAM processes during learning; (2) SRL is context-specific and therefore successful learning may require a learner to increase/decrease the use of certain key

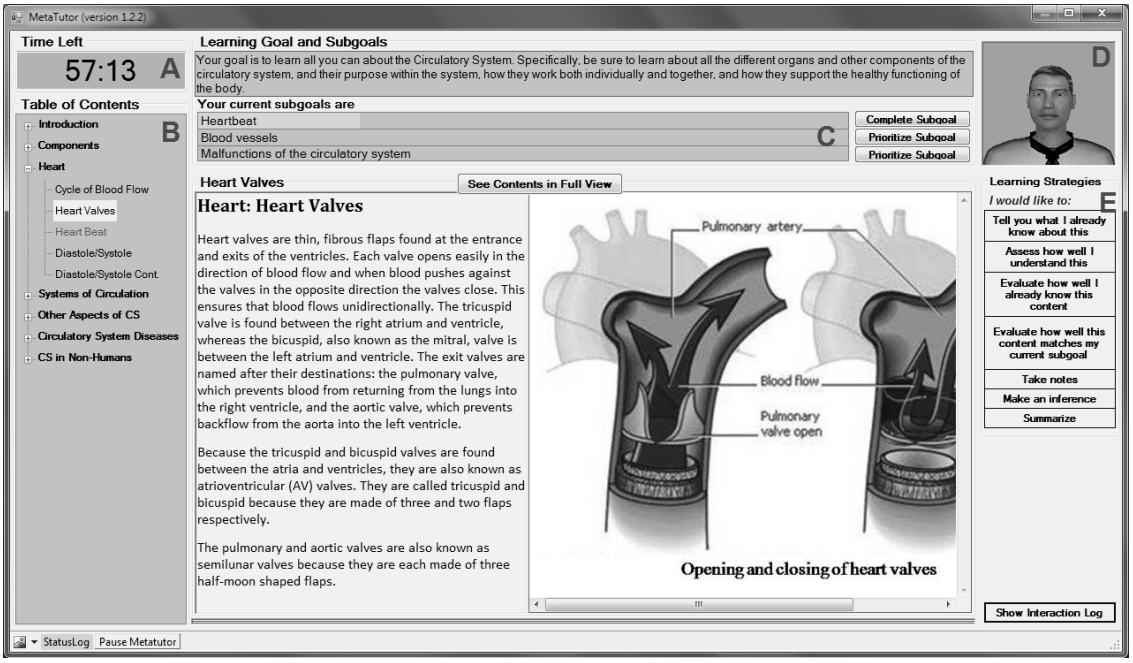



Figure 1: Annotated screenshot of MetaTutor (A: time remaining in the session, B: table of contents, C: current subgoals and progression, D: embodied pedagogical agent, E: palette of monitoring and strategy actions)

SRL processes at different points in time during learning; (3) a learner's ability to monitor and control both internal (*e.g.*, prior knowledge) and external factors (*e.g.*, changing dynamics of the learning environment; relative utility of an agent's prompt) are crucial in successful learning; (4) a learner's ability to make adaptive, real-time adjustments to internal and external conditions, based on accurate judgments of their use of CAM processes, is fundamental to successful learning; and; (5) certain CAM processes (*e.g.*, interest, self-efficacy, task value) are necessary to motivate a learner to engage and deploy appropriate CAM processes during learning and problem solving. This model is best suited for this project since it deals specifically with the person-in-context perspective and postulates that CAM processes occur during learning with a multi-agent system, which will be useful in examining when and how learners will regulate their learning about the human circulatory system. As such, the macro-level processes used in this paper are reading, metacognitive monitoring, and learning strategies. Reading behavior is critical since it is the most important activity related to acquiring, comprehending, and using content knowledge related to the science topic. During reading, learners need to monitor and regulate several key processes such as: (1) selecting relevant content (*i.e.*, text and diagrams) based on their current sub-goal; (2) spending appropriate amounts of time on each page, depending on their relevance regarding their current sub-goal; (3) deciding when to switch or create a new sub-goal; (4) making accurate assessments of their emerging understanding; (5) conceptually connecting content with prior knowledge; (6) adaptively selecting, using, and assessing the effective use of several learning strategies including re-reading, coordinating informational sources, summarizing, making inferences, in order to comprehend the material at various levels (*i.e.*, declarative, procedural, and conceptual knowledge); and, (7) making adaptive changes to behavior based on a variety of external (*e.g.*, quiz scores, quality and timing of agents' prompts and feedback) and internal sources (*e.g.*, affective experiences including both positive and negative affective states, perception of task difficulty). In sum, SRL involves the continuous monitoring and regulation of CAM processes during learning with MetaTutor.

## 3.3 Participants and data collection

While data has been collected over a sample of 148 undergraduate students from two large public universities in North America, we consider for this study only a sub-sample of 51 participants from the experimental condition that included the most prompts from the pedagogical agents to perform SRL actions and in which students were given some adaptive feedback after having performed those actions. Participants from other conditions did not receive a similar experience with the system, and therefore the values of the variables considered (*cf.* section **??**) were completely different for them (*e.g.* they took less quizzes as they were not prompted to self-regulate their learning). Considered logs contained an average of 1072 events per session ($\sigma = 255$).

# 4. PRELIMINARY STEPS

## 4.1 Data preparation, coding and extraction

For the analysis performed here, as justified in section **??**, we abstracted the set of collected interactions into three broad categories: reading, monitoring, and strategy (*cf.* Table **??** for The detailed list of actions extracted from the data).

### *4.1.1 Reading*

A reading action (*Read*) is coded each time the student clicks to display a new page of content to read. They can be split

according to two combinatorial criteria, $r$ and $t$, written as $Read_t^r$, where:

- $r$ stands for the relevance of the page with regard to the student's current subgoal (+ for a relevant page, − for an irrelevant page, ∅ if no subgoal is currently set and relevance can't be determined);
- $t$ stands for the time the student spent reading the page ($s$ if they remain less than 15 seconds, threshold under which no SRL prompt can be triggered, $l$ otherwise).

#### 4.1.2 Monitoring

A monitoring action ($Mon$) is coded when the student performs, or is prompted to perform, a monitoring action with respect to their learning. This monitoring action could be a judgment of learning (JOL) about what they have just read, a feeling of knowing (FOK) regarding the content of the page, an evaluation of the content (CE) relevance with respect to their current subgoal, or an assessment of their progress towards their current subgoal (MPTG). They can also be split according to two combinatorial criteria, $e$ and $i$, written as $Mon_i^e$, where:

- $e \in \{+, -, \varnothing\}$ stands for the correctness of the monitoring evaluation performed by the student (+ if the evaluation is right, − if it is wrong, ∅ if no direct evaluation is possible for the monitoring process);
- $i \in \{u, a\}$ stands for the initiator of the action ($u$ for the user, $a$ for the agent).

Following FOKs and JOLs, as well as when the student claims to have finished a subgoal, students are asked to answer a short quiz (of 3 to 10 questions). Those actions, coded as $Quiz$, can be split along one dimension and are then written $Quiz^s$, where $s \in \{+, -\}$ stands for the success or failure to pass the test (+ if the student obtained more than 66% of correct answers, − otherwise).

#### 4.1.3 Strategy

A strategy action ($Str$) is coded when the student uses a strategy to self-regulate their learning, including when the strategy is prompted by the agent, as well as when the user independently decides to perform the action. Strategy actions include a summarization (SUMM) of the page, a coordination of information sources (COIS) by viewing a related image, an inference (INF) regarding the reading material, a re-reading (RR) of a paragraph that was not well understood, or notes taken about the reading material. This action can also be split depending on the initiator of the action, and is then written $Str_i$, where $i \in \{u, a\}$ as defined in 4.1.2. Moreover, we distinguish a particular strategy consisting of taking or checking notes in the embedded note interface or using the electronic paper-based notepad provided next to the workstation. These note actions are coded as $Notes$.

### 4.2 User clustering

#### 4.2.1 Methodology

In a previous study [**?**], we ran a cluster analysis over a subset of 13 variables extracted from the interaction log after the end of the student's learning session: pretest and posttest score, number of subgoal and page quizzes, mean first score in subgoal and page quizzes, proportion of subgoals attempted among the 7 possible, number of subgoals changes, total session duration, time spent reading content, number of times the student took notes and checked notes, and the duration of the note-taking episodes. This analysis empoyed the Expectation-Maximization (EM) algorithm as implemented in the Weka data mining package [**?**]. The number of categories to find being undetermined a priori, we used a 10-fold cross-validation, during which we incremented the number of clusters (starting with 1) as long as the loglikelihood averaged over the 10 folds was increasing (*i.e.* we stopped as soon as the loglikelihood with N+1 clusters was lower than with N clusters). We used 1000 different initialization seeds for the EM algorithm, in order to compensate for its tendency to get stuck into local optima, and selected, among the 1000 partitions of students generated, the most frequent one among the most frequently obtained number of clusters (3).

Table 2: Synthesis of clusters differences (italic means clusters weren't significantly different from one another according to that variable when using an ANOVA with $p < 0.05$)

| Variables | Score for each cluster | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| Pretest score | M | L | H |
| Posttest score | M | L | H |
| *Session duration* | *M* | *M* | *M* |
| Reading duration | H | H | L |
| Proportion of subgoals attempted | L | L | H |
| Number of subgoals changes | M | L | H |
| *Number of subgoals quizzes* | *M* | *M* | *M* |
| Mean first score in subgoal quizzes | M | L | H |
| *Number of page quizzes* | *M* | *M* | *M* |
| Mean first score in page quizzes | M | L | H |
| Number of note taking | H | L | L |
| Number of note checking | L | L | H |
| Time spent taking notes | H | L | L |

#### 4.2.2 Results

Three clusters were obtained, which characteristics are summarized in Table **??**, where clusters 0, 1 and 2 are made of 21, 14 and 16 students, respectively. Generally, students from cluster 2 scored high on pretest, posttest and intermediary quizzes, spent less time than others reading while attempting more subgoals, and took less notes and less time taking them. In contrast, students from cluster 1 scored low on pretest, posttest and intermediary quizzes, attempted less subgoals and took few notes and less time to take them. Students from cluster 0 occupied generally a intermediate position in terms of performance and subgoal uses, but took overall more notes and more time to take them. When using a formula derived from [**?**] to evaluate learning gains (*cf.* [**?**] for more details), we also found that students from cluster 2 had the most significant knowledge acquisition, as opposed to those in cluster 1. For all those reasons, cluster 1 will

Table 1: List of actions extracted from MetaTutor interaction logs

| Category | Action name | Description |
|---|---|---|
| Read | $Read_s^+$ | Student skims through a page relevant for their current subgoal for less than 15s |
| | $Read_s^-$ | Student skims through a page irrelevant for their current subgoal for less than 15s |
| | $Read_s^\varnothing$ | Student skims through a page without having a subgoal set for less than 15s |
| | $Read_l^+$ | Student reads a page relevant for their current subgoal for more than 15s |
| | $Read_l^-$ | Student reads a page irrelevant for their current subgoal for more than 15s |
| | $Read_l^\varnothing$ | Student reads a page without having a subgoal set for more than 15s |
| Monitoring | $Mon_a^+$ | Student is prompted to evaluate their knowledge, learning or the relevance of the content they are reading, and evaluates correctly |
| | $Mon_a^-$ | Student is prompted to evaluate their knowledge, learning or the relevance of the content they are reading, and is wrong in their evaluation |
| | $Mon_a^\varnothing$ | Student is prompted to perform a monitoring action that doesn't require an evaluation |
| | $Mon_u^+$ | Student takes the initiative of evaluating their knowledge, learning or the relevance of the content they are reading, and evaluates correctly |
| | $Mon_u^-$ | Student takes the initiative of evaluating their knowledge, learning or the relevance of the content they are reading, and is wrong in their evaluation |
| | $Mon_u^\varnothing$ | Student takes the initiative of performing a monitoring action that doesn't require an evaluation |
| | $Quiz^+$ | Student passes a page or subgoal quiz (more than 66% of correct answers) |
| | $Quiz^-$ | Student fails a page or subgoal quiz (less than 66% of correct answers) |
| Strategy | $Str_a$ | Student is prompted to deploy a strategy to self-regulate |
| | $Str_u$ | Student takes the initiative of using a strategy to self-regulate |
| | $Notes$ | Student takes or checks notes using the embedded interface or a paper-based electronic notepad |

be referred to as cluster L (for low), cluster 2 as cluster H (for high) and cluster 0 as cluster M (for medium). The fact that exactly three (as opposed to any other number) clusters were extracted might sound unsurprising, but comes from the fact that it was the best partition of the subjects in the 13-dimension space considered.

# 5. DIFFERENTIAL SEQUENCE MINING

## 5.1 Method principles

To identify important activity patterns in a comparison between student clusters, we employ a differential sequence mining technique [**?**]. This technique uses sequence mining and two different measures of pattern frequency to identify *differentially frequent* patterns between two sets of action sequences. Differential sequence mining combines frequency measures and techniques from sequential pattern mining [**?**], which determines the most frequent action patterns across a set of action sequences, and episode mining [**?**], which determines the most frequently used action patterns within a given sequence.

The sequential pattern mining frequency measure (*i.e.*, how many sequences/students exhibit the given pattern) is used to identify patterns common to a group of students. We refer to this as the "sequence support" (*s-support*) of the pattern, and we call patterns meeting a given s-support threshold *s-frequent*. In this analysis, we employ an s-support threshold of 0.5 to focus on patterns exhibited by at least half of a given group of students. The episode mining frequency (*i.e.*, the frequency with which the pattern is repeated *within* an action sequence) is important for assessing the extent to which a student relies on a particular pattern of activities. For a given student, we refer to this as the "instance support" (*i-support*), and we call patterns meeting a given i-support threshold *i-frequent*. To calculate the i-support of a pattern for a group of students, we use the mean of the pattern's i-support values across all traces in the group.

The differential sequence mining technique first uses a sequential pattern mining algorithm to identify the patterns that meet a minimum s-support constraint within each group [**?**]. To compare the identified frequent patterns across groups, we calculate the i-support of each pattern for each student (in each group). Using a t-test, we filter the s-frequent patterns to identify those for which there is a statistically significant difference in i-support values between groups. Comparing the mean i-support value for each pattern between groups then allows us to focus the comparison on patterns that are employed significantly more often by one group than the other.

This comparison produces four distinct categories of frequent patterns: two categories where the patterns are s-frequent in only one group, illustrating patterns primarily employed by the respective groups, and two categories where the patterns are common to both groups but used significantly more often in one group than the other. The patterns in each of these qualitatively distinct categories are (separately) sorted by the difference in mean group i-support[1] to focus the analysis on the most differentially frequent patterns [**?**].

## 5.2 Application to the data

In order to identify patterns more closely related to changes in students' knowledge and understanding, we decided to focus mainly on clusters H and L, as defined in section 4.2.2. Moreover, to further identify the patterns most characteristic of students in cluster H (*resp.* L) we identified differentially frequent patterns with respect to the other two

[1]Even though a pattern may not be s-frequent in a group of action sequences, it can still occur in some sequences in the group, so an i-support value can be calculated (or the i-support is 0 if the pattern does not occur in any trace in the group).

clusters M and L (*resp.* M and H) in a secondary analysis. We employed an s-support threshold of 50% in this analysis, to consider all the patterns that were exhibited by at least half of the students in a given cluster, and a standard value of 0.05 for the t-test cutoff $p$ value. We tried to preliminarily group sequences of identical actions together, but the results obtained were not very different from the ones without grouping, as the data extracted do not display long sequences of similar actions – therefore, those results are not reported here. Similarly, although we also considered the possibility of using gaps of one or more actions when identifying patterns, we discarded this analysis because the frequency of events collected in the log is low, which means that even a gap of only one action could mean that two actions of a pattern are actually separated by a rather long period of inactivity.

## 5.3 Results

The Table **??** displays the patterns with the highest difference of S-support between clusters H and L (positive value in column 3) as well as between clusters L and H (negative value in column 3), provided that difference is statistically significant (*i.e.* a t-test $p$ value below 0.05 in column 4). It also displays a selection of interesting patterns, which differed in a statistically significant way between the two clusters. Columns 6 to 11 provide the results obtained for that selection of patterns using two different samples of students: first (columns 6 to 8), cluster H alone and a merge of clusters L and M, and then (columns 9 to 11), cluster L alone and a merge of cluster H and M. Columns 5, 8 and 11 show, for the two considered samples, if only one or both of them were having a s-support above 50% for the considered pattern. Values N/A are used when the pattern is non statistically significant for the two considered samples.

The following observations can be made:

– According to *pattern 1*, when prompted to use a strategy (regardless of the one suggested by the agent), students in cluster H reacted by taking notes more often than students in cluster L. We already knew that students in cluster H had received significantly more prompts from the system, and taken less notes overall than those in cluster L (but checked them more often). This pattern seems to suggest that the reason might be that the notes they were taking mainly came from prompts from the agents. Moreover, since when they type a summary, students are offered the possibility to add it to their notes, it appears that students from cluster H must have preferred that strategy, which also would explain why they spent less time with the note-taking interface open (since the summary is typed in a different text box, and the note-taking interface is opened only to add the already typed text). Finally, the fact that the difference for this pattern is significant for cluster H *vs.* L, H *vs.* M&L and H&M *vs.* L indicate that the degree to which one relies on the prompt for notes or summaries to take notes is directly correlated to the belonging to one of the three clusters (*i.e.* this behavior is observed more in cluster H than in M, and more in M than in L).

Similarly, *pattern 3* indicates that after a note-taking event, students from cluster H often moved on to another relevant page, which they read for an extended period. *Pattern 5*, which is a combination of patterns 1 and 3, confirm the idea that students from cluster H had a very methodical approach to navigating through the content: they selected a relevant page, read it until being prompted by the agent to take notes or summarize it, performed that action, and then moved on to a new relevant page. Incidentally, it also indicates their effectiveness in identifying a page relevant to their current subgoal simply from its title (since that is all they can see before opening it). This latter hypothesis is itself reinforced by the observation that *patterns 10 and 11*, relative to a brief visit on an irrelevant page or to a succession of brief visits to irrelevant pages, is characteristic of students from clusters M and L, as opposed to students from cluster H who seem to not even need to open the pages to figure out they are irrelevant to their current subgoal.

– *Pattern 2* simply confirms what we already knew about the tendency of student in cluster H to have answered correctly more often to intermediate quizzes (for a page or a subgoal). It also significantly distinguish members of cluster H from those in cluster M&L considered together.

– *Patterns 4 & 7* are relative to pages viewed when the students did not have any active subgoal set. Pattern 4 indicates that students in the cluster H have visited more pages for a long time without having a subgoal set, which is confirmed by pattern 7 which also indicate an alternation between short and long reads when no subgoals were set. As we also know that students from cluster H attempted more subgoals overall than students in the cluster L, it cannot mean that they have simply refused to set additional subgoals once they had finished their original ones (e.g., in an attempt to get rid from the system prompts and feedback), but rather that: a) they might have spent some time reviewing pages already read before taking the posttest, and/or b) instead of setting a final subgoal when they did not have much time left, they took some time to review the pages they had not yet explored.

This hypothesis can be confirmed by looking at the temporal distribution of those two patterns: for students in cluster H, the median time is of 108 and 112 minutes (for an overall session of approximately 120 minutes), which means that it's during the last 15 minutes of their learning session that students were displaying that kind of browsing behavior, clearly distinct from the ones they had displayed earlier in the session.

– *Pattern 6* indicates that students in cluster H seemed to more often estimate properly their level of understanding of the content or the relevance of the page they were visiting when it was relevant for their current subgoal. While this pattern is only marginally significant when comparing clusters H and L, it is statistically significant when comparing H to M&L, confirming that it is specific of students in cluster H. It tends to show that not only other students had difficulties to identify the relevance of a page from its title, but that even once they had been able to spend some time reading its content, they were less prone to correctly evaluate its relevance or their understanding of it.

This hypothesis seems to be confirmed by the complementary *pattern 8*, which indicates that students from cluster L, when they were on a page irrelevant for their subgoal for a long time and got prompted to evaluate its relevance (the only prompt they can get on a non-relevant page), tended to be wrong in their evaluation.

If we consider again the temporal distribution of those two patterns, we can notice that the median time, for students in cluster L, is of 50 and 45 minutes, *i.e.* less than the median time of the session (60 minutes). We can therefore assume

Table 3: Significant and most frequent patterns differentiating clusters

| # | Pattern | Cluster H vs. L | | | Cluster H vs. M&L | | | Cluster H&M vs. L | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | I-Supp. Diff | t-test (p value) | S-Freq. Cluster | I-Supp. Diff | t-test (p value) | S-Freq. Cluster | I-Supp. Diff | t-test (p value) | S-Freq. Cluster |
| 1 | $Str_a \prec Notes$ | **3.93** | **0.002** | **Both** | **3.28** | **0.005** | **Both** | **2.30** | **0.007** | **Both** |
| 2 | $Quiz^+$ | **3.10** | **0.036** | **Both** | **2.09** | **0.046** | **Both** | 2.30 | 0.086 | Both |
| 3 | $Notes \prec Read_l^+$ | **2.86** | **0.004** | **Both** | **2.35** | **0.012** | **Both** | **1.71** | **0.012** | **Both** |
| 4 | $Read_l^{\varnothing}$ | **2.63** | **0.039** | **H** | **2.27** | **0.050** | **H** | 1.48 | 0.107 | H&M |
| 5 | $Str_a \prec Notes \prec Read_l^+$ | **2.38** | **0.001** | **Both** | N/A | N/A | N/A | **1.27** | **0.017** | **Both** |
| 6 | $Read_l^+ \prec Mon_a^+$ | 1.96 | 0.065 | Both | **1.96** | **0.048** | **Both** | 0.85 | 0.304 | Both |
| 7 | $Read_s^{\varnothing} \prec Read_l^{\varnothing}$ | **1.33** | **0.050** | **H** | 1.23 | 0.061 | H | N/A | N/A | N/A |
| ... | | | | | | | | | | |
| 8 | $Read_l^- \prec Mon_a^-$ | **-0.54** | **0.039** | **L** | N/A | N/A | N/A | -0.25 | 0.360 | L |
| 9 | $Read_s^+ \prec Read_l^-$ | **-0.65** | **0.012** | **L** | N/A | N/A | N/A | **-0.53** | **0.038** | **L** |
| 10 | $Read_s^- \prec Read_s^-$ | N/A | N/A | N/A | **-1.77** | **0.030** | **M&L** | N/A | N/A | N/A |
| 11 | $Read_s^-$ | -3.49 | 0.149 | Both | **-2.56** | **0.036** | **Both** | -2.39 | 0.321 | Both |

that, at least, students from cluster L have been slightly improving their capacity to evaluate their learning and the relevance of a page over time.

– *Pattern 9* confirms the previous observation that students in cluster L really had issues to see the relevance of a page with regard to their subgoal: they did not simply end up going to random pages that were irrelevant to their subgoal, or ignored the subgoal they had set, but instead, they appeared to sometimes skim through a relevant page, miss its relevance, and end up instead spending a long time on a page that wasn't irrelevant to their subgoal. This tendency is shared, to some extent, with students from cluster M, as the results of clusters H *vs.* M&L are also statistically significant.

– A final observation can be made regarding the tendency of a student to obey system prompts: if we run the same analysis without distinguishing the correctness of the evaluation of students monitoring (*i.e.* by considering actions $Mon_a = Mon_a^+ \cup Mon_a^-$ and $Mon_u = Mon_u^+ \cup Mon_u^-$), we observe that the pattern $Mon_a \prec Mon_u$is significantly more frequent for students in cluster H, which tends to indicate that when prompted to perform an optional monitoring action (most likely, a MPTG, since otherwise there should be a *Quiz* action following the $Mon_a$), they are more prone to accomplish the suggested action.

## 6. DISCUSSION

To summarize the results obtained in the previous section, we can conclude that students from cluster H are more inclined to follow the system prompts and to follow the suggestions to take notes or summarize what they have just learned. Further, they are more prone to keep applying the same method for each page they read, are better at identifying a page relevant to their subgoal from its title, and are more strategic in their preparation for the posttest (*e.g.*, they usually use their last 10 to 15 minutes to briefly review various pages). From an ITS design point of view, the fact that these students used system prompts to effectively regulate their learning tends to indicate that the frequency of Strategy prompts should probably not be reduced. However, as they seem good at distinguishing relevant pages from irrelevant ones, they might need less scaffolding regarding the Monitoring processes. On the other hand, students from cluster L appear particularly unable to identify pages relevant to their subgoal, which is probably linked to their lower prior knowledge. For them, it seems that additional scaffolding from the system would certainly be beneficial. However, even when prompted to monitor their learning, they tend to be mistaken in their evaluation. Therefore, it could be necessary to go further than the methods currently employed to suggest ways in which they can better evaluate the relevance of a page.

## 7. CONCLUSION, FUTURE DIRECTIONS

In this paper, we have presented a two-step analysis of data collected with an ITS designed to foster self-regulated learning. First, the clustering of students using Expectation Maximization has allowed us to distinguish three clusters of students with different prior knowledge on the topic, learning performance, and strategies. We then described a set of actions extracted from the system interaction trace log and employed a sequence mining technique to identify differentially frequent activity patterns. We used the identified patterns to characterize students from different clusters with particular emphasis on those that had the highest and the lowest learning gains. We have been able to identify patterns of actions that suggest students with high prior knowledge and high learning gains tended to be more compliant with system prompts, using them to validate their progression. Further, these students were better at identifying pages relevant to their subgoals from the page title and tended to have a phase at the end of the session during which they reviewed the content in preparation for the posttest.

The analysis performed here will allow us to more accurately identify the group to which a student belongs during their use of MetaTutor and dynamically adapt the scaffolding and feedback mechanisms accordingly. Another future research direction will involve the use of other channels of data collected while students use MetaTutor (eye-tracking information, affective data extracted from video captures, and think-aloud data) in order to enhance our identification and understanding of phases when low-performing students are unable to properly monitor their learning.

## 8. ACKNOWLEDGMENTS

This research was supported by NSF-IIS award #0904387 to the third author and NSF-DRL awards (#0633918 and

#1008282) to the fourth author.

## 9. REFERENCES


[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh IEEE International Conference on Data Engineering (ICDE)*, pages 3–14, 1995.

[2] R. Azevedo, R. Behnagh, M. Duffy, J. M. Harley, and G. J. Trevors. Metacognition and self-regulated learning in student-centered leaning environments. In D. Jonassen and S. Land, editors, *Theoretical foundations of student-center learning environments*, pages 216–260. Erlbaum, Mahwah, NJ, 2nd edition, 2012.

[3] R. Azevedo, J. G. Cromley, D. C. Moos, J. A. Greene, and F. I. Winters. Adaptive content and process scaffolding: A key to facilitating students' self-regulated learning with hypermedia. *Psychological Testing and Assessment Modeling*, 53:106–140, 2011.

[4] R. Azevedo and R. Feyzi-Behnagh. Dysregulated learning with advanced learning technologies. *Journal of e-Learning and Knowledge Society*, 7(2):9–18, 2011.

[5] R. Azevedo, A. Johnson, A. Chauncey, and A. C. Graesser. Use of hypermedia to convey and assess self-regulated learning. In B. Zimmerman and D. Schunk, editors, *Handbook of self-regulation of learning and performance*, pages 102–121. Routledge, New York, 2011.

[6] R. Azevedo, D. C. Moos, A. Johnson, and A. Chauncey. Measuring cognitive and metacognitive regulatory processes used during hypermedia learning: Issues and challenges. *Educational Psychologist*, 45(4):210–223, 2010.

[7] R. Azevedo and A. M. Witherspoon. Self-regulated learning with hypermedia. In D. J. Hacker, J. Dunlosky, and A. C. Graesser, editors, *Handbook of metacognition in education*, pages 319–339. Routledge, Mahwah, NJ, 2009.

[8] F. Bouchet, J. M. Harley, G. J. Trevors, and R. Azevedo. Clustering and profiling students according to their interactions with an intelligent tutoring system fostering Self-Regulated learning. *Journal of Educational Data Mining*, (submitted), 2012.

[9] S. D'Mello, B. Lehman, J. Sullins, R. Daigle, R. Combs, K. Vogt, L. Perkins, and A. Graesser. A time for emoting: When Affect-Sensitivity is and isn't effective at promoting deep learning. In V. Aleven, J. Kay, and J. Mostow, editors, *Intelligent Tutoring Systems*, volume 6094, pages 245–254. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[11] J. S. Kinnebrew, K. M. Loretz, and G. Biswas. A differential sequence mining method contextualized by student performance evolution to derive learning behavior patterns. *Journal of Educational Data Mining*, (submitted), 2012.

[12] H. Mannila, H. Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[13] R. Martinez, K. Yacef, J. Kay, A. Al-Qaraghuli, and A. Kharrufa. Analysing frequent sequential patterns of collaborative learning activity around an interactive tabletop. In *Proceedings of the Fourth International Conference on Educational Data Mining.*, Eindhoven, Netherlands, 2011.

[14] J. C. Nesbit, Z. Mingming, Y. Xu, and P. H. Winne. Advancing log analysis of student interactions with cognitive tools. In *12th Biennial Conference of the European Association for Research on Learning and Insruction (EARLI)*, Budapest, Hungary, 2007.

[15] D. Perera, J. Kay, I. Koprinska, K. Yacef, and O. R. Zaane. Clustering and sequential pattern mining of online collaborative learning data. *IEEE Trans. on Knowl. and Data Eng.*, 21(6):759–772, June 2009.

[16] C. Romero, S. Ventura, A. Zafra, and P. d. Bra. Applying web usage mining for personalizing hyperlinks in web-based adaptive educational systems. *Computers & Education*, 53(3):828–840, Nov. 2009.

[17] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Advances in Database Technology - EDBT'96*, volume 1057, pages 1–17. Springer-Verlag, Berlin/Heidelberg, 1996.

[18] T. Y. Tang and G. McCalla. Student modeling for a web-based learning environment: a data mining approach. In *Proc. of the 18th national conference on Artificial intelligence*, page 967968, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[19] P. H. Winne. Self-regulated learning viewed from models of information processing. In B. Zimmerman and D. Schunk, editors, *Self-regulated learning and academic achievement: Theoretical perspectives*, pages 153–189. Erlbaum, Mahwah, NJ, 2001.

[20] P. H. Winne and A. Hadwin. Studying as self-regulated learning. In D. J. Hacker, J. Dunlosky, and A. C. Graesser, editors, *Metacognition in educational theory and practice*, pages 277–304. Erlbaum, Hillsdale, NJ, 1998.

[21] P. H. Winne and A. Hadwin. The weave of motivation and self-regulated learning. In D. Schunk and B. Zimmerman, editors, *Motivation and self-regulated learning: Theory, research and applications*, pages 297–314. Routledge, 2008.

[22] B. Zimmerman and D. Schunk, editors. *Handbook of self-regulation of learning and performance*. Routledge, New York, 2011.

# Learner Differences in Hint Processing

Ilya M. Goldin
Human-Computer Interaction Institute
Carnegie Mellon University
goldin@cmu.edu

Kenneth R. Koedinger
Human-Computer Interaction Institute
Carnegie Mellon University
koedinger@cmu.edu

Vincent Aleven
Human-Computer Interaction Institute
Carnegie Mellon University
aleven@cs.cmu.edu

## ABSTRACT

Although ITSs are supposed to adapt to differences among learners, so far, little attention has been paid to how they might adapt to differences in how students learn from help. When students study with an Intelligent Tutoring System, they may receive multiple types of help, but may not comprehend and make use of this help in the same way. To measure the extent of such individual differences, we propose two new logistic regression models, ProfHelp and ProfHelp-ID. Both models extend the Performance Factors Analysis model (Pavlik, Cen & Koedinger, 2009) with parameters that represent the effect of hints on performance on the same step on which the help was given. Both models adjust for general student proficiency, prior practice on knowledge components, and knowledge component difficulty. Multilevel Bayesian implementations of these models were fit to data on student interactions with a geometry ITS, where students received on-demand problem-relevant help ranging from first-level hints that facilitate application of principles to specific and immediately actionable bottom-out hints. The model comparison showed that in this dataset students differ in their individual hint-processing proficiency and these differences depend on hint levels. These results suggest that we can assess specific learning skills, e.g., making sense of instructional text, and in future work we may be able to remediate and improve such skills.

## Keywords

Effect of help on performance, individual differences, learning skills, multilevel Bayesian models, Item Response Theory

## 1. INTRODUCTION

In virtually all imaginable learning settings, when students work through problems, they may seek help. But are all students able to benefit from help equally, and are there meaningful differences across types of help?[1]

Our long-term goal is to answer this and other questions related to the nature of the learning skills that students bring to bear when working with educational technologies, as well as whether or not there are significant individual differences in these learning skills. Seeking help and learning from help [1, 19] may be one set of such learning skills, which can include both the metacognitive monitoring needed to determine when soliciting help benefits learning, as well as making sense of instructional text in the context of problem solving. If individual differences in learning skills exist, and if they can be assessed, an Intelligent Tutoring System may be able to adapt to these differences, to provide students with appropriate metacognitive support, and perhaps even to improve learning skills.

[1] This work is supported in part by Postdoctoral Training Grant awarded to Carnegie Mellon University by the Department of Education (#R305B110003).

In this project, we aim to determine whether or not there are significant variations in students' abilities to make use of help. As a first step, we examine how well students can use help to solve the task at hand (i.e., the problem step they are working on). While the effect of help on learning, rather than performance, is of primary long-term interest, if a student cannot make good use of help "locally" (on the current step), it is unlikely such help will enhance learning (i.e., enhance performance on a future related task). [11] In other words, studying the "local" effect of help on performance is useful, because any beneficial effect of help on performance may be a harbinger of longer-term effect on learning.

Specifically, our research questions are: How well do students perform after receiving hints, and does performance after hints differ across hint levels? Are there individual differences in how effective hints are among students, and if so, are the individual differences consistent within each student across hint levels? Are the individual differences, if any, related to general student proficiency in solving problems?

We analyze data generated in the course of another study, and use statistical methods to account for potentially confounding variables, including general student proficiency, prior practice on knowledge components, and knowledge component difficulty.

One prior effort to evaluate the effect of hints on same-item performance is by the developers of the Mastering physics ITS. In [12], a 2PL Item Response Theory model was fit to performance on first attempts, after which separate models were fit to each of several paths through the ITS. Unlike that effort, our work examines individual differences with various types of help, and addresses potential confounds due to variability in prior practice and due to difficulty of knowledge components (rather than just unique problem items). We also analyze a larger dataset, and fit parameters relating to various types of help simultaneously in a Bayesian Markov Chain Monte Carlo (MCMC) framework to account for uncertainty during estimation.

Mining data from the Geometry Cognitive Tutor (an earlier version of the tutor whose data is analyzed in the current study), we showed that asking for help is beneficial for local performance. [1] Specifically, asking for help after one or two errors on a step was compared to attempting to solve the step again. Asking for help, compared to continued trying, was associated with fewer subsequent errors on the given step and a reduction in the time needed to complete the step. However, [1] did not look into individual differences in students' ability to take advantage of help to improve performance on problem steps, and did not investigate differences between hint levels.

Another related study [7] presents two models, a learning decomposition and an extension to Bayesian Knowledge Tracing. The latter is particularly interesting in that it aims to distinguish the effect of help as a performance scaffold from its effect on learning. However, neither model addresses multiple hint levels or individual differences in hint-processing proficiency.

**Table 1: Examples of hint messages**

| Knowledge Component | First Hint | Second Hint | Third Hint |
|---|---|---|---|
| Triangle-Sum-Answer | In this problem, you have triangle SOL. You know the measure of two of the angles in this triangle, namely, angles DSO and OLD. | The sum of the measures of the interior angles of a triangle is 180 degrees. | m∠SOL = 180 - m∠DSO - m∠OLD. |
| Triangle-Sum-Reason | In this problem, you have Triangle WAR. You know the measure of two of the angles in this triangle, namely, angles ARO and OWA. | The sum of the measures of the interior angles of a triangle is 180 degrees. | You can find the measure of Angle WAR by applying the "Triangle Sum" theorem. |
| Separate-Complementary-Angles-Answer | The problem statement says that angles ∠XSD and ∠JNT are *complementary angles*. | Complementary angles are angles whose measures add up 90 degrees. | m∠XSD = 90 - m∠JNT. |
| Angle-Addition-Answer | Angles DGF and MGD are *adjacent angles*. This means that they share a side (namely, GD) but do not overlap. Together they form ∠MGF. | When an angle is formed by two or more adjacent angles, the measure of that angle is equal to the sum of the adjacent angles. Therefore, m∠MGF = m∠DGF + m∠MGD. | [No third level hint.] |

An exploratory analysis of our dataset (Section 2) shows that selection effects confound a naïve approach that merely tallies successful and unsuccessful performance with and without hints. Section 3 proposes two logistic regression models that take these confounds into account. Section 4 describes the results of fitting multilevel Bayesian implementations of these models to our data. The final sections discuss the results, limitations, contributions of this research, and future directions.

## 2. EXPLORATORY DATA ANALYSIS

The study that produced the dataset analyzed here took place at a vocational school [17]. Three 9th grade classes of 51 participating students, led by the same teacher, used Geometry Cognitive Tutor as part of regular instruction about twice a week for five weeks. Students worked through problems, most of which contained multiple steps. There were 170 distinct problems, consisting of 1666 problem steps. Problems were assigned to students according to a mastery criterion based on the Knowledge Tracing [8] algorithm in the Cognitive Tutor software, i.e., each student only saw a subset of the 170 problems.

Using this software, a student may make multiple attempts to complete a problem step. Completing a step requires a correct response; giving a correct response on the first attempt means that this student will never see a hint. On each attempt, a student may supply a correct answer, an incorrect answer, or may ask for a hint. The first hint that the student sees is called "help level 1", the second is "help level 2", and so on to the final ("bottom-out") hint, which in our dataset is help level 3 or 4. (Table 1) For students who do not know how to respond, the bottom-out hint often states exactly what the response must be.

In general, a first hint points out relevant problem features, and it defines key terms, e.g., "vertical angles." Second hints state the problem-solving principle that is applicable given the features pointed out in the first hint, in terminology consistent with the first hint. Third hints derive an expression for the sought angle measure (in terms of known angle measures). Using this expression, the angle measure can be found in a straightforward manner, by first substituting in the values for the angle measures referenced in the expression, and then evaluating the resulting arithmetic expression. The rationale for sequencing hint levels from less specific to more specific was to try to give the student as much opportunity as possible to "generate" the step, which may include retrieving a relevant problem-solving principle, as discussed in [4] and [3].

Interaction with such hint sequences may lead some students (e.g., those who are relatively less proficient) to request hints more often than others. Similarly, some problem steps (e.g., those that are challenging) may lead to hint requests relatively more often.

As a measure of student proficiency, we consider how often a student responds correctly to a problem step on the first attempt. Specifically, a crude measure of proficiency is the success rate on first attempts, i.e., the proportion of all problem steps that the student answered correctly on first attempt out of all those first attempts where a student gave a correct or an incorrect response (omitting first attempts where the student requested a hint).

Given this measure, is proficiency related to use of hints? For each student, the hint use rate is the proportion of problem steps on which this student requested one or more hints out of all attempted problem steps. The correlation of student proficiency and hint use rate is $r$=-0.84, i.e., hints are more likely to be requested by less proficient students.

Similarly, as a measure of problem step difficulty, we take the proportion of first attempts on the step to which a student gives a correct response out of correct and incorrect first attempts (again, omitting first attempts that are hint requests). Is step difficulty related to use of hints? The rate of hint use on a problem step is the proportion of students who request any hints on the step out of all students who attempt the step. The correlation of step easiness (1 - step difficulty) and rate of hint use is $r$=-0.68, i.e., hints are more likely to be requested on steps that are harder.

Do hints of different levels differ in their effect on performance? If requesting a hint counts as unsuccessful performance (Table 2, top row), the success rate drops from first attempts (78%) to attempts after first and second hints (21% and 37%). However, when students request a first hint, the next action that they are most likely to perform in the tutor is to ask for a second hint (87% of the time). Students ask for a third hint as the likely next action after the second (88% of the time). Not counting hint requests (Table 2, bottom row), performance after the first hint (68%) is lower than after the second and third hints (83% and 88%).

**Table 2: Success rates after hints, counting Correct, Incorrect, and Hint outcomes**

| Success Rate Formula | On First Attempt | After 1st Hint | After 2nd Hint | After 3rd Hint |
|---|---|---|---|---|
| $C/(C+I+H)$ | 78% | 21% | 37% | 82% |
| $C/(C+I)$ | 83% | 68% | 83% | 88% |

To sum up this exploratory analysis, we find that hints are more likely to be requested by less proficient students; hints are more likely to be requested on steps that are difficult; and success after first hints is less likely than after second and third hints.

The exploratory analysis is appealing, but possibly misleading. First, what is "student proficiency"? A student who is proficient may simply have had more opportunities to practice the relevant skills, which would cause a selection effect for this analysis, or there may be additional differences in student ability that cannot be observed directly. Second, while an ITS may tutor all students on the same skills, it may assign students different problems. If so, skills rather than problem steps would be the right grain size for analysis. Third, since students see different problems, and problems involve different hints, there could be selection effects in terms of how we measure performance after hints for different students. Thus, it would be desirable to control for proficiency, prior practice, selection effects related to problem difficulty, and to take into account a model of skills in the domain. As described in the following section, we can use a logistic regression to take these elements into account.

# 3. METHODS

We fit two models to these data, both extending the Performance Factors Analysis (PFA) model. [14] PFA is a logistic regression that is fit to correct and incorrect student responses.

$$logit(\Pr(Y=1)) = \sum_{j \in KC} (\beta_j + \gamma_j s_{i,j} + \rho_j f_{i,j})$$

**Equation 1: Performance Factors Analysis (PFA) model**

Under PFA, the probability of a correct response by a pupil on a problem step, i.e., of $Y = 1$, is determined by a linear combination of parameters related to the knowledge components (KCs) that are thought to be relevant to that step. Parameter $\beta_j$ denotes the easiness of $KC_j$. Parameters $\gamma_j$ and $\rho_j$ are weights on the observed frequency of successful ($s_{i,j}$) and unsuccessful ($f_{i,j}$) prior practice by the same learner $i$ on the same KC $j$. The innovation in PFA was to separate $\gamma_j$ and $\rho_j$, the effects of successful and unsuccessful prior practice, rather than collapsing these effects as one parameter.

**Table 3: Example of instances in our dataset**

| Pupil | Item | Attempt | Prior Practice | Outcome |
|---|---|---|---|---|
| 5 | Prob1.St3 | 1 | $S_{5,9}$=3; $F_{5,9}$=1 | First hint |
| 5 | Prob1.St3 | 2 | $S_{5,9}$=3; $F_{5,9}$=1 | Incorrect |
| 5 | Prob1.St3 | 3 | $S_{5,9}$=3; $F_{5,9}$=1 | Correct |

Our interest is in learner performance in the presence of help on attempts after the first. The original use of PFA was to model unassisted performance; in PFA, the outcome variable $Y$ and the prior practice counts $s_{i,j}$ and $f_{i,j}$ only represent first attempts on a problem step, not subsequent attempts. By contrast, we fit our models to outcomes both at first attempts and at each attempt that was the next action after a hint (but the prior practice counts still represent only first attempts).

Consider the example in Table 3, where a student (pupil 5) makes three attempts on the same item (problem 1, step 3). When the ITS initially presents the student with this step in the course of solving the problem, the student requests a hint. This hint is at the first of several levels of help (usually 3 or 4) that the ITS may offer on a problem step. According to the knowledge component model for the problems in this dataset, this step has a single relevant KC with KC id=9. This student has had prior practice opportunities with this KC: three were successful, and one was not. Counts of prior practice are based on first-attempts only; thus, when this student practices this KC on a future item, prior practice counts will be $S_{5,9}$=3; $F_{5,9}$=2, because the outcome of the first attempt in this example was unsuccessful. This example yields two instances to be input to the logistic regression, corresponding to the first 2 attempts. Both attempts are coded as having the outcome 0 (only correct outcomes are coded as 1). For the purpose of estimating the help-level parameters in our model, the first-attempt instance is coded as not following a help message, and the second-attempt instance is coded as following a hint at help level 1. We assume that the effect of a hint should be observable in the next attempt on the same step. Because attempt 3 follows an input rather than a hint display, its outcome is not directly attributable to a hint, and this attempt does not yield an instance. Of the 28777 transactions in this dataset, 17515 were first attempts, 4466 attempts were the next action after some kind of a hint, and the rest were not entered as instances because they were not next actions following a hint.

The first model, ProfHelp, examines how help levels differ in their effect on performance, but does not consider individual differences in hint processing among students.

$$logit(\Pr(Y=1)) = \theta_p + \lambda_h + \sum_{j \in KC} (\beta_j + \gamma_j s_{i,j} + \rho_j f_{i,j})$$

**Equation 2: Proficiency and Help (ProfHelp) model**

The innovation in this model is the $\lambda_h$ parameter. One $\lambda_h$ is fit for every attempt after a hint. (Because help may be requested as a first attempt, but never prior to a first attempt, $\lambda_0 = 0$.) One of $\lambda_1, .., \lambda_4$, respectively, represents the contribution of having just seen a first, second, third or fourth hint to the probability of successful performance on this subsequent attempt. Another view of $\lambda_h$ is that it represents average proficiency in processing level-*h* hints. Parameters other than $\lambda_h$ control for student proficiency, problem step difficulty via a decomposition on knowledge components, and prior practice on knowledge components. In other words, the effect of having just seen a hint is not confounded by the findings that hints are more likely to be requested by less proficient students and on more difficult items (Section 2), nor by the intuition that a lack of prior practice can lead to more frequent hint requests. Finally, one $\theta_p$ parameter, as in Item Response Theory (IRT) models, is fit for every student *p*, representing the baseline proficiency of that student.

$$logit(\Pr(Y=1)) = \theta_p + \lambda_{p,h} + \sum_{j \in KC} (\beta_j + \gamma_j s_{i,j} + \rho_j f_{i,j})$$

**Equation 3: ProfHelp-ID (Individual Differences) model**

The second model, ProfHelp-ID, considers that the same help level may have different effects on different students. The

difference from the ProfHelp model is in the $\lambda_{p,h}$ parameter, where the subscripts $p, h$ indicate that a separate parameter is fit for each pupil in each help level. This represents the pupil's individual hint-processing proficiency. These parameter estimates are pooled across pupils within a single help level via a multilevel model (bold typeface denotes hyperparameters):

$$\lambda_{p,h} \sim N(\boldsymbol{\lambda}_h, \boldsymbol{\sigma}_h^2)$$

For instance, the ProfHelp-ID model stipulates that $\lambda_{p,2}$, i.e., each per-pupil estimate of the effect of responding after a second hint (h=2) is drawn from a distribution with mean $\boldsymbol{\lambda}_2$ and variance $\boldsymbol{\sigma}_2^2$ that is shared across pupils. In this way, information on each pupil helps determine a baseline effect of seeing a second hint, and the baseline effect helps constrain the estimate of the per-pupil individual differences.

Partial pooling is appropriate for this problem not only for statistical parsimony, but also because it lets us be conservative in making a claim about the presence of individual differences. (Partial pooling is similar to the idea of a random effect, where values are assumed to come from a broader sample of interest, rather than a fixed effect, where all values of interest are represented.) The alternative, a no-pooling model, would treat pupils as independent of one another. This means that first, the no-pooling model could detect individual differences even when the differences are small (i.e., not meaningful), and second, unpooled individual differences would be hard to quantify because there may be very few observations for any particular pupil at a given help level. The partial pooling pulls all individual difference estimates towards the mean, reducing the effect of small differences, and it helps compensate for data sparsity by using the hyperparameter estimates as prior information for the parameters. (Note that model ProfHelp is the complete-pooling version of ProfHelp-ID, in that ProfHelp does not allow for individual differences in hint processing.)

The models were fit using the JAGS software for Bayesian modeling [15], which is an effective platform for fitting Item Response Theory and similar models (e.g., [9]). For each model, we ran 4 sampling chains, with 400 adaptation iterations (discarded). Inferences below are based on every 10th draw (thinning) of 1000 iterations. Model convergence and mixing across chains were verified by visual examination of autocorrelation, trace and density plots.

# 4. RESULTS

As multilevel Bayesian models, ProfHelp and ProfHelp-ID may be compared in terms of Deviance Information Criterion (DIC). DIC is similar to AIC in that it rewards models that fit the data well but penalizes an increase in the number of parameters in the model. [16] DIC takes into account that in Bayesian models with pooling, the effective number of parameters is itself estimated as a posterior distribution of a random variable.

**Table 4: Model-fitting results**

| Model | Deviance | Effective Parameters | DIC |
|---|---|---|---|
| ProfHelp | 22013 | 135 | 22149 |
| ProfHelp-ID | 21741 | 220 | 21962 |

As Table 4 shows, the ProfHelp-ID model is preferable to the ProfHelp model on this dataset in that the improvement in prediction accuracy outweighs the increase in the number of parameters. Relative to the ProfHelp approach of fitting a single parameter across all students within a help level (complete pooling), the partial-pooling approach of ProfHelp-ID finds that there are individual differences in student performance after a hint at each help level. This finding is despite the fact that ProfHelp-ID is nonetheless more conservative than a no-pooling model.

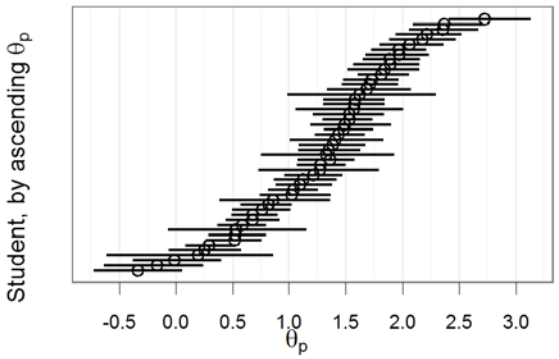



**Figure 1: Medians and 95% CI for $\theta_p$ under ProfHelp-ID**

The $\theta_p$ proficiency parameter (Figure 1) is positive for most of the students, reflecting the prevalence of successful first attempts in this dataset (the model predicts that a student for whom $\theta_p = 0$ will answer correctly on 50% of first attempts, given that the other terms are zero). The $\theta_p$ parameter is entered into the model for both first attempts and later attempts, and both could affect its estimate. However, first attempts are much more frequent than later ones, and $\lambda_{p,h}$ provides an intercept for each pupil on the later attempts. This effectively makes $\theta_p$ a constant baseline for $\lambda_{p,h}$ that is unaffected by the later attempts.[2]

ProfHelp-ID measures the effect of having seen a hint on the immediately preceding attempt as a baseline across all students (the $\lambda_h$ hyperparameter), and as a deviation from this baseline for every pupil, $\lambda_{p,h}$. The improved fit of ProfHelp-ID over ProfHelp implies that the mean effects $\lambda_h$ are correct only on average, not for all students. As Figure 2 shows, the mean effect, in logit units, of having just seen a hint (solid black vertical line in each of the three frames) are approximately -2.4, -1.7 and 0.5 for first, second and third help levels, respectively.[3] These differences are significant, as indicated by the non-overlapping 95% credible intervals (grey vertical lines on the left and right of each black line). The mean effects of first and second hints are negative, which implies that, on average, the performance of all students, proficient or not, and on all problem steps, easy or difficult, is lower after these hints than would be predicted based only on overall proficiency $\theta_p$. The effect of third hints is only somewhat

[2] A parameter in a logistic regression adds to the model's estimate of success on a given instance. To interpret a coefficient, a rule of thumb is to divide by 4. For example, if $\theta_1 = 2$, that adds 0.5 to the probability that model will predict success on every attempt by pupil 1.

[3] As a check on the model fitting, the estimates of $\lambda_{p,h}$ from ProfHelp were similar, -2.3, -1.5, and 0.5. There were few observations for performance after a fourth hints, so we omit discussion of $\lambda_{p,4}$ and $\lambda_4$.

positive. Converted to probabilities, effects of first and second hints at -2.4 and -1.7 logit units, respectively, implies that a student with median proficiency on this dataset ($\theta_p = 1.4$), on a problem step of average difficulty ($\sum_{j \in KC} \beta_j = 0$), and with no prior practice on relevant KCs, is predicted to respond correctly 27% of the time after first hints and 42% of the time after second hints. These predicted correctness rates are higher than those of the "naïve" analysis (21% and 37%, Table 2) that does not take into account proficiency and other confounds. While these rates are low, they are nonetheless an improvement over the students' failures to answer correctly on the first attempt.

An unexpected finding is that general proficiency $\theta_p$ is *negatively* correlated with hint-processing proficiency $\lambda_{p,h}$: for first, second, and third hints, $r$=-0.48, $r$=-0.54, and $r$=-0.41, $p < 0.01$ for all. The more proficient the student, the less likely it is that the student benefits from a hint. This relationship is also visible in Figure 2, where each frame is ordered by ascending $\theta_p$. Hint-processing proficiency of first hints $\lambda_{p,1}$ is also correlated with hint-processing proficiency of second hints $\lambda_{p,2}$, $r$=0.34, $p$<0.05; other hint-processing proficiencies are uncorrelated with each other.

## 5. DISCUSSION

We aimed to understand the nature of learning skills such that we can support learning more effectively. We found that hints levels differed in their effect on performance, and only level-1 and level-2 hint-processing proficiencies correlated with each other. Further, there were individual differences in hint-processing proficiency, and general proficiency was negatively correlated with hint-processing proficiency.

Given how hint levels are implemented (Table 1), it is not surprising to see better performance on the next attempt after the bottom-out hint level, compared to the next attempt after other hint levels. As mentioned, all that correct performance following a bottom-out hint requires is algebraic substitution and arithmetic, which are likely to be mastered skills for our student population. By contrast, correct performance after first and second hint levels requires interpretation of mathematical text that refers to potentially unmastered geometry concepts and principles. To solve problems in the geometry unit in this dataset, one needs to retrieve a general geometry principle, to apply the principle to the problem by mapping it to specific problem features, and to perform algebra and arithmetic according to the principle. Before the principle can be retrieved, salient problem features need to be identified. Level-1 hints tend to point out the salient problem features and define key terms. Level-2 hints state what principle is applicable given the salient features pointed.

The negative effects of level-1 and level-2 hints are consistent with prior work on hint effectiveness. [1] As pointed out in [7], "students request help on [items] on which they have low knowledge. The help thus acts as *evidence* of a lack of knowledge, rather than a direct *cause* of that lack of knowledge." Further, neither short nor long hint reading times are positively associated with learning. [18] Another explanation for the negative coefficients for our dataset in particular is that the logistic regression is effectively forced to estimate these very negative effects given the prevalence of positive $\theta_p$ values (which are in turn due to the prevalence of successful first attempts).

Prior work suggests that it can be fruitful to consider how tutor behavior may differentially affect students across varying levels of KC mastery. [2] The ProfHelp models are based on the

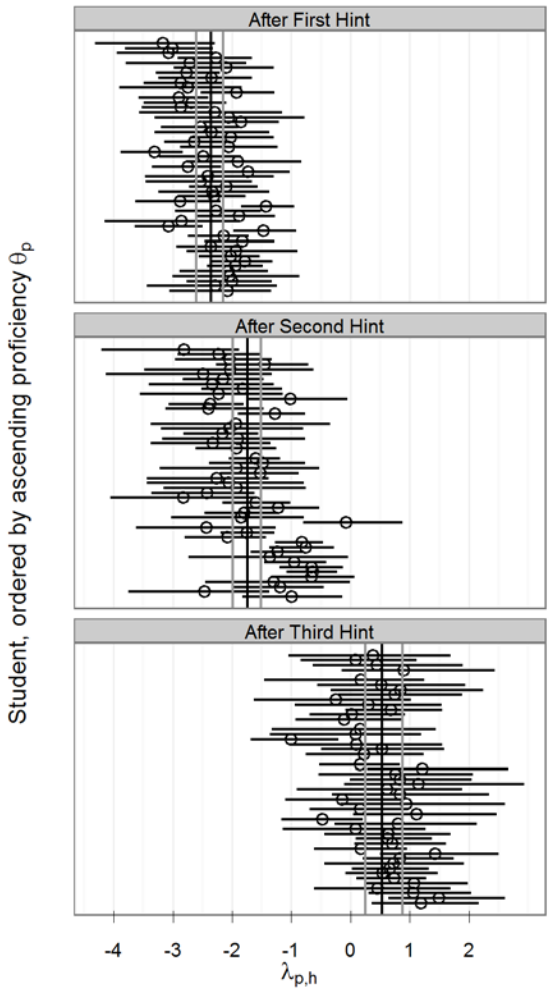



**Figure 2: Medians and 95% CI for $\lambda_{p,h}$ under ProfHelp-ID; median of $\lambda_h$ (black vertical) and 95% CI (grey vertical)**

psychometric concept of latent traits that is inherent to Item Response Theory. IRT models are said to be unidimensional if they represent proficiency with one parameter $\theta_p$ per student. ProfHelp-ID relaxes this unidimensionality assumption via parameters $\lambda_{p,h}$ per student for attempts after hints, but retains it within each type of attempt (first attempt and after each hint level). Thus, the dimensions of proficiency in ProfHelp-ID (first attempts and help levels) may not represent proficiency ideally. The ProfHelp-ID estimate of the probability of success will be in error when performance within this type of attempt is multidimensional, e.g., if an otherwise easy KC unexpectedly challenges a proficient student (or, vice versa, if a student with low proficiency succeeds quickly on a generally difficult KC). Having found individual differences within different attempt types, we speculate as to the nature of the learning skills that may be involved in interpreting hints and using them to support correct performance. This analysis will inform future model refinements.

**Success after level-1 hint with good knowledge of relevant KC.** A student who is close to KC mastery did not succeed on the first attempt on a step, but did on the next action after a level-1 hint. The failure on the first attempt may have been an "identification

slip", i.e., a slip in identifying the relevant problem features that was due to random circumstance rather than lack of knowledge, or to high cognitive load such as could be expected in a dataset of quite complex geometry problems that involve multiple steps and multiple problem-solving principles. Level-1 hints point out problem features that are relevant to the application of a principle, but not what principle to use, or how. Because the student succeeded after the level-1 hint, the student was apparently able to retrieve and apply the principle (i.e., did not need further hints) once given the salient features, but required assistance to identify the salient features. When hints are used to fix "identification slips," no hint interpretation skills are needed; the hint serves as reminder of something the student already knows but failed to retrieve. The student still applied "principle application skills" to the extent that the knowledge of how to apply this principle had not yet been proceduralized or automated.

**Success after level-1 hint with little knowledge of relevant KC.** By contrast, an identification slip is not possible for a student with little knowledge of the relevant KC. Given that level-1 hints state problem features relevant to the application of a principle, success after a level-1 hint suggests that this hypothetical student was able to infer a correct answer from a set of problem features, even without knowing the rule that connects the features to the answer. Perhaps Assuming this was not a guess, the student induced a valid principle from the given example, and then used principle application skills mentioned above, though a less generous interpretation would suggest that the student learned shallowly. Quite an impressive feat of unsupervised inductive learning, with less than a single example to work with and no outcome given! How could this be possible? Perhaps this student drew on additional information sources, e.g., student peers or the textbook. Perhaps the diagram helps; e.g., once one sees a visual representation of adjacent angles, the notion that the measure of an angle made up of adjacent angles is the sum of the two measures of the adjacent angles seems quite intuitive. Other geometry knowledge may help as well. For instance, smart students may be able to infer the vertical angles theorem from the linear pair postulate.

**Success after level-2 hint with good knowledge of relevant KC.** Failure after a level-1 hint followed by success after a level-2 hint suggests that the student needed to be reminded of the relevant domain principle. This student should have been able to retrieve the relevant domain principle from memory given the prior practice of the KC. What could cause failure to retrieve a principle? Similar to failure on a first attempt, one cause may be a mere "applicability slip" in mapping problem features to a known principle, e.g., due to random occurrence or to overwhelming cognitive load. Another cause may be that there is a phase in the normal skill acquisition process in which students have more trouble recognizing the applicability of rules than in applying them once cued to critical problem features. In other words, while in this phase, students need to be reminded of a principle, but can apply it, especially when also given some key information (as in the level-1 hint) on how to instantiate the principle. This hypothesized phase also explains failure after the level-1 hint.

The modest but statistically significant correlation of $\lambda_{p,1}$ and $\lambda_{p,2}$ suggests that the two hint levels may be linked in how they affect students, but that there are also some differences. One explanation for the correlation is that level-1 and level-2 hints would both be skipped by a student engaged in "help abuse" [19], causing both level-1 and level-2 hints to be associated a 0 (unsuccessful) logistic regression outcome. By contrast, bottom-out hints cannot be skipped, so unsuccessful outcomes after bottom-out hints would not be confounded with help abuse. Another cause for the correlation may well the requirement, shared across the level-1 and level-2 hints, to apply a principle, while the requirements of bottom-out hints, likely mastered by all students, would not induce a correlation. Finally, the two hint levels may share the hypothesized phase affecting students with good KC knowledge.

One difference between level-1 and level-2 is that answering correctly after (only) a level-1 hint requires more domain-specific knowledge than answering correctly after a level-2 hint. One way to answer correctly after a level-1 hint is to retrieve the relevant problem-solving principle from memory, possibly cued by the problem features pointed out in the hint, and to apply the rule successfully, helped perhaps by the information provided in the hint. By contrast, to answer correctly after a level-2 hint, it is not necessary to retrieve the principle from memory, since the level-2 hint provides a statement of the principle. The student must still do some work to figure out how the rule applies.

An instance of poor retrieval may be symptomatic of a broader retrieval deficiency on the part of the student, which would constitute a learning skill deficiency. Success after first hints occurred frequently enough (predicted 27% correctness rate for a student with median general proficiency) that it may be worth investigating whether such a deficiency could be detected, or even addressed. Ideally, learners could be supported in overcoming such a cognitive shortcoming on their own. Students need to apply general principles to specific problems in many domains (e.g., [10]), and it would be interesting to see if such a skill could transfer.

**Success after level-2 hint with little knowledge of relevant KC.** Poor retrieval cannot explain success after a level-2 hint when a student has had little prior practice on the relevant KC, i.e., when there is no expectation for retrieval. A level-2 hint states the rule that applies, but not *how* it applies. Thus, such successful performance may indicate that the student is skilled at applying a somewhat unfamiliar problem-solving principle, when given a statement of that principle (level-2 hint) and key problem features that instantiate the principle's applicability conditions (level-1 hint). What remains for the student to do is still rather involved: apart from understanding the principle, the principle has to be mapped onto the problem, a process that (facilitated by the level-1 hint) requires dealing with difficult terminology in relating the general terms in which the rule is stated to the specific problem.

**Success after level-3 hint.** With respect to the level-3 hints, it seems unlikely that correct performance after a bottom-out hint involves important learning skills, aside from possibly a general tendency to carefully follow very specific instructions.

In sum, this analysis contemplates several hypothesized metacognitive skills. Success after level-1 and level-2 hints for a student with high KC knowledge may indicate deficiencies in identifying salient problem features, mapping a principle to salient features, and retrieving a principle. Success after level-2 hints for a student with little KC knowledge may indicate skill in applying unknown principles (i.e., parsing and mapping—with some help—of an unfamiliar principle). Our results could be viewed as implying that different students possess these different learning skills to different degrees. This interpretation addresses both possible causes of differences between hint levels and possible causes of learners' differences in hint processing. For instance, if

we could find a way to help students learn to recognize when a geometry principle applies, this should both improve the effectiveness of first hints, and reduce unexplained variability among students in terms of their hint-processing proficiency. Perhaps if students were given instruction to look for diagram features that can cue a principle, then on a first hint like "The problem statement says that angles ∠XSD and ∠JNT are *complementary angles*", they might be better able to interpret the notion of complementary angles by paying attention to that part of the diagram. The analysis considered the major findings that hint levels differ in their effect on performance, that student proficiencies with level-1 and level-2 hints are modestly correlated, and that there are individual differences in hint-processing proficiency.

Finally, we address the finding that general proficiency is negatively correlated with hint-processing proficiency. One explanation is that this finding is merely an artifact induced by the statistical model. In designing the ProfHelp-ID model, we reasoned that to ascribe an effect to some proficiency with hints we had to partial out the effect of general proficiency. In fact, making $\theta_p$ a baseline for $\lambda_{p,h}$ may overcorrect for any relationship between general proficiency and hint proficiency. The linear combination of the two parameters effectively subtracts $\theta_p$ from $\lambda_{p,h}$, which means that $\lambda_{p,h}$ contains information on $\theta_p$, and that can induce the negative correlation.[4] While such a correlation complicates interpretation of parameter estimates, it would not invalidate the model fit. A second explanation is that individuals with a higher proficiency may be less proficient with hints because they have less practice using them.

One contribution of the ProfHelp models is that they control for selection effects due to general student proficiency, prior practice on knowledge components, and knowledge component difficulty. The models here do not account for other selection effects, which we intend to address in future work. First, ProfHelp treats all hint messages at a given level as equally effective, while messages associated with different KCs may in fact have differential effects on student performance. (Such an analysis would be the "KC differences" analogue of the individual differences analysis presented here.) In this way, we might be able to identify specific hint messages that are significantly more or less effective than other messages to inform ITS design. Second, in future work we intend to relax the unidimensional IRT assumption, i.e., to handle the case that a KC that the model estimates to be easy may challenge a student that the model estimates to be proficient. Third, the ProfHelp models do not account for patterns of use that students may follow. For instance, in discussing the effect of level-1 and level-2 hints it would be desirable to account for the effects of help abuse, e.g., a student clicking through the hints without reading them. [2, 6] The ProfHelp models do not distinguish such behavior from spending a long time on each hint, which may indicate deliberative reflection.

The need for future research is highlighted by the ProfHelp-ID estimates of effectiveness of hints: 27% and 42% accuracy after level-1 and level-2 hints, respectively, for a student with median general proficiency. While even these relatively low levels of effectiveness improve, by definition, over the students' failure to answer correctly on the first attempt, there is clearly room to make hints more effective, and hence a need for research on hints types and hint processing. The ProfHelp-ID model may serve as a tool for such research. Given that this model can fit transaction data from an ITS, one can expect to apply it again in the future to evaluate alternative hinting strategies.

# 6. CONCLUSIONS

The results presented here may be said to pose more questions than they answer, which is appropriate for an early project in a relatively unexplored area. Significantly, the results show that hints levels differ in their effect on performance, and that there are individual differences in hint-processing proficiency. These findings account for general student proficiency, prior practice on knowledge components, and knowledge component difficulty via the ProfHelp and ProfHelp-ID models. The next steps are to understand the causes of the individual differences, and to try to detect them automatically.

An additional contribution of this research is the new Bayesian implementation of the new ProfHelp and ProfHelp-ID models (and by extension, the PFA model).[5] The flexibility of the JAGS modeling tool is well-suited to logistic regressions such as these and to the need for rapid prototyping of model variations. The time saved in development easily outweighs potentially slow MCMC sampling. Moreover, the model-fitting process can easily be parallelized for separate MCMC chains.

This research has wide impact. The data analyzed here come from a system in the Cognitive Tutor family, in use by over 600,000 students. [5] The same methods would apply to any software that uses either progressive hint sequences or multiple independent types of help. For instance, in SQL Tutor, "an error flag message informs the student about the clause in which the error occurred. A hint-type message gives more information about the cause of error. Partial solution feedback displays the correct content of the clause in question, while the complete solution simply displays the correct solution of the current problem." [13] The Masteringphysics ITS includes three types of hints ("a list of steps, declarative statements, and procedural subtasks") and other types of help. [12]

Among the limitations of this research, the first is that it considers the effect of hints on performance, not learning. As [7] points out, in theory, a hint may both scaffold performance on the current step and it may teach the student in preparation for a subsequent problem. However, as evidenced by the analysis in Section 5, while the effects on learning are important, the effects on performance are not yet well understood.

Other limitations are due to the assumptions embedded in the PFA model and the ProfHelp models. These include that knowledge components are independent and linearly additive; that the effects of the problem step are fully represented by the relevant knowledge components and prior practice on these KCs; and that there are no problem effects, e.g., steps within the same problem are treated as independent of one another. The ProfHelp models are limited in that they only consider the effect of help from the immediately preceding attempt, while there could be effects that carry over from earlier attempts. In the dataset examined here, hint levels were always presented in the same order, and the differential effects of hint types could not be teased apart using

[4] To see how this would work, let $X$ and $Y$ be two independent random normal variables. Let $X' \leftarrow Y - X$. By definition, $cor(X,Y) = 0$, but $cor(X',Y) \sim 0.71$.

[5] Please contact the corresponding author for the JAGS code.

ProfHelp. However, this is a limitation of the dataset rather than ProfHelp itself.

In future work, we plan to extend the ProfHelp models. We may incorporate students' hint-level preferences, such as to take into account the tendency of some students to click through to the bottom-out hint without making attempts after first and second hints. We may also incorporate the number of prior hint episodes on practice opportunities of various KCs to distinguish the effect of prior hints from the effect of incorrect prior performance.

At the same time, regression techniques cannot eliminate all selection effects. Future work should include controlled experiments that compare different hint types, and an evaluation of their effects on learning and on reduction of unexplained variance in hint processing among students.

## 7. ACKNOWLEDGMENTS

The authors thank Brian Junker and Georg Goerg for advice on statistical modeling, and S. McKay Curtis for addressing many questions about the mcmcplots package.

## 8. REFERENCES

[1] Aleven, V. and Koedinger, K.R. 2000. Limitations of Student Control: Do Students Know when They Need Help? *Intelligent Tutoring Systems* (Berlin, Heidelberg, 2000), 292–303.

[2] Aleven, V., Mclaren, B., Roll, I. and Koedinger, K. 2006. Toward meta-cognitive tutoring: A model of help seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence in Education*. 16, 2 (2006), 101–128.

[3] Aleven, V., Stahl, E., Schworm, S., Fischer, F. and Wallace, R. 2003. Help Seeking and Help Design in Interactive Learning Environments. *Review of Educational Research*. 73, 3 (Sep. 2003), 277–320.

[4] Anderson, J.R. 1993. *Rules of the Mind*. Psychology Press.

[5] Apollo Group to Acquire Carnegie Learning: 2011. *http://carnegielearning.com/press-room/press-releases/2011-08-02-apollo-group-to-acquire-carnegie-learning/*. Accessed: 2012-04-16.

[6] Baker, R.S., Corbett, A.T., Koedinger, K.R. and Wagner, A.Z. 2004. Off-task behavior in the cognitive tutor classroom: when students game the system. *Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), 383–390.

[7] Beck, J.E., Chang, K., Mostow, J. and Corbett, A. 2008. Does Help Help? Introducing the Bayesian Evaluation and Assessment Methodology. *Intelligent Tutoring Systems*. B.P. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, eds. Springer Berlin Heidelberg. 383–394.

[8] Corbett, A.T. and Anderson, J.R. 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*. 4, 4 (1995), 253–278.

[9] Curtis, S.M. 2010. BUGS Code for Item Response Theory. *Journal of Statistical Software*. 36, 1 (2010), 1–34.

[10] Goldin, I.M., Pinkus, R.L. and Ashley, K.D. accepted. Validity and Reliability of an Instrument for Assessing Case Analyses in Bioengineering Ethics Education. *Science and Engineering Ethics*.

[11] Kirschner, P.A., Sweller, J. and Clark, R.E. 2006. Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*. 41, 2 (Jun. 2006), 75–86.

[12] Lee, Y.-J., Palazzo, D., Warnakulasooriya, R. and Pritchard, D. 2008. Measuring student learning with item response theory. *Physical Review Special Topics - Physics Education Research*. 4, 1 (2008).

[13] Mitrović, A. 1998. Experiences in implementing constraint-based modeling in SQL-Tutor. *Intelligent Tutoring Systems* (1998), 414–423.

[14] Pavlik Jr, P., Cen, H. and Koedinger, K. 2009. Performance Factors Analysis - A New Alternative to Knowledge Tracing. *14th International Conference on Artificial Intelligence in Education* (Brighton, England, 2009), 531–538.

[15] Plummer, M. 2003. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing* (Vienna, Austria, 2003).

[16] Plummer, M. 2008. Penalized loss functions for Bayesian model comparison. *Biostatistics*. 9, 3 (Jul. 2008), 523–539.

[17] Salden, R.J.C.M., Aleven, V.A.W.M.M., Renkl, A. and Schwonke, R. 2008. Worked Examples and Tutored Problem Solving: Redundant or Synergistic Forms of Support? *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (Austin, TX, 2008).

[18] Shih, B., Koedinger, K. and Scheines, R. 2010. Unsupervised discovery of student learning tactics. *Proceedings of the Third International Conference on Educational Data Mining* (2010).

[19] Wood, H. and Wood, D. 1999. Help seeking, learning and contingent tutoring. *Computers and Education*. 33, 2 (1999), 153–170.

# Methods to find the number of latent skills

Behzad Beheshti
Département de génie informatique et génie logiciel
Polytechnique Montréal
C.P. 6079, succ. Centre-Ville
Montréal, Québec, Canada
behzad.beheshti@polymtl.ca

Michel C. Desmarais
Département de génie informatique et génie logiciel
Polytechnique Montréal
C.P. 6079, succ. Centre-Ville
Montréal, Québec, Canada
michel.desmarais@polymtl.ca

Rhouma Naceur
Département de génie informatique et génie logiciel
Polytechnique Montréal
C.P. 6079, succ. Centre-Ville
Montréal, Québec, Canada
rhouma.naceur@polymtl.ca

## ABSTRACT

Identifying the skills that determine the success or failure to exercises and question items is a difficult task. Multiple skills may be involved at various degree of importance, and skills may overlap and correlate. In an effort towards the goal of finding the skills behind a set of items, we investigate two techniques to determine the number of dominant latent skills. The Singular Value Decomposition (SVD) is a known technique to find latent factors. The singular values represent direct evidence of the strength of latent factors. Application of SVD to finding the number of latent skills is explored. We introduce a second technique based on a *wrapper* approach. Linear models with different number of skills are built, and the one that yields the best prediction accuracy through cross validation is considered the most appropriate. The results show that both techniques are effective in identifying the latent factors over synthetic data. An investigation with real data from the fraction algebra domain is also reported. Both the SVD and *wrapper* methods yield results that have no simple interpretation.

## 1. INTRODUCTION

A critical component of student models is the skills mastery profile. Personalization of the learning content relies heavily on this component in many, if not most intelligent tutoring systems. The more precise the skills mastery profile is, the more appropriate this personalization process will be.

However, finding the latent skills underlying exercises or questions items is non-trivial because of a number of reasons.

One reason is that multiple skills may be involved at various degree of importance with regards to a single item. This is in fact typical of most items. For example solving a simple fraction algebra problem may require knowledge of a few algebra rules, each rule representing a specific skill. More general skills such as vocabulary and grammar rules may be involved in language related task, etc.

Another difficulty is that skills may overlap and they will therefore correlate. Highly correlated skills result in similar response patterns to a set of items.

Finally, the nature of the items and the difficulty of mastering some skills will result in *slip* and *guesses*. Those will be reflected as noise that will make the identification of the latent skills more difficult.

Most of the time, the latent skills underlying question items are defined by experts. Models such as Knowledge Tracing [2], Constraint-based Modeling [7], or Performance Factor Analysis [8], are well known examples that require expert defined mapping of skills to latent factors. Some studies have looked at means to help this process.

Suraweera et al. have used an ontology-based approach to facilitate the item to skill mapping and the more general task of building the domain model [9].

Others have studied the mapping of items to skills with data driven algorithms with some success [1; 3; 11]. Their results show that mappings can be successfully derived in certain conditions of low noise (*slip* and *guess*) relative to the latent factors. However, these studies assume that the number of skills are known in advance, which is rarely the case. Although some of the the latent skills may be relatively obvious, the obvious skills only set a minimum number. That minimum does not preclude that other skills may come into play and have a strong effect also.

Of course, we do not need to identify all the skills behind an item in order to use the item outcome for assessment purpose. As long as we can establish a minimally strong tie from an item to a skill, this is a sufficient condition to use the item in the assessment of that skill. But knowledge that there is a fixed number of determinant factors to predict item outcome is a useful information. For example, if a few number of skills, say 6, are meant to be assessed by a set of 20 questions items, and we find that the underlying number of determinant latent factors behind these items is very different than 6, then it gives us a hint that our 6-skills model may not be congruent with the assessment result.

This study aims at identifying this number. It aims at finding means to estimate how many latent factors are influencial enough to determine the item success. We explore two techniques towards this end: Singular Value Decomposition (SVD) and a *wrapper* selection feature based on Non-negative Matrix Factorization (NMF). We describe these techniques in more details and report the results of our experiments to validate their effectiveness for estimating the number of latent skills[1].

[1]The reader interested in more details is referred to the code that was used in this study: `http://www.professeurs.polymtl.ca/michel.desmarais/Papers/EDM2012/scripts.html`

## 2. SVD-BASED METHOD

Singular Value Decomposition (SVD) is a well known matrix factorization technique that decomposes any matrix, $\mathbf{A}$, into three sub-matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T \quad (1)$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and their column vectors respectively represent the eigenvectors of $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$. $\mathbf{D}$ is a diagonal matrix that contains the singular values. They are the square root of the eigenvalues of the eigenvectors and are sorted in a descending order.

Because the singular values represent scaling factors of the unit eigenvectors in equation (1), they are particularly useful in finding latent factors that are dominant in the data. This is demonstrated with simulated data below. First we describe the simulated data and the results of applying SVD on the students item outcome results matrix $\mathbf{R}$.

### 2.1 Simulated data

The synthetic data is generated by defining a Q-matrix of 21 items that combine 6 skills. The 21 items are represented as columns in figure 1. They span the space of all pairwise combinations of skills (first 15 columns) plus 6 single skill items (last 6 columns).

| Skills | Items |
|---|---|
| 1 | 111110000000000100000 |
| 2 | 100001111000000010000 |
| 3 | 010001000111000001000 |
| 4 | 001000100100110000100 |
| 5 | 000100010010101000010 |
| 6 | 000010001001011000001 |

Figure 1: Conjunctive Q-matrix composed of 21 items that span all combinations of 6 skills for pairs of skills and single skills

Figure 1's Q-matrix is used to generate simulated data and we assume a *conjunctive* model (all skills are necessary to answer the item correctly). The data contains the 21 question items and 200 simulated student responses over these items. The six skills are assigned an increasing degree of difficulty from 0.17 to 0.83 on a standard normal (Gaussian) scale, and each student is assigned a skill vector based on a {0,1} sampling with a probability corresponding to this difficulty (or *easiness* in fact, since higher values bring greater chances of skill mastery). The choice of these difficulty values stems from the need to have a mean student success score around 50%–60%: because 15 of the 21 items require the conjunction of two skills, mean skill mastery must be substantially higher than 50% to obtain average results around 50%–60%.

Once a skills mastery profile is assigned to students, represented by a matrix $\mathbf{S}$, an *ideal response matrix* is generated according to the product $\neg\mathbf{R} = \mathbf{Q}\neg\mathbf{S}$, where $\mathbf{Q}$ is a *conjunctive* Q-matrix (more details about this model are given later, see equation (3) below). Then, slip and guess factors are used to generate noise in the ideal response pattern by randomly changing a proportion of the item success and failures outcomes according respectively to slip and guess values. The slip and guess values of respectively 0.1 and 0.2 will result in approximately 15% of the item outcomes being inconsistent with the ideal response matrix (15% corresponds to a weighted average of 0.1 and 0.2).

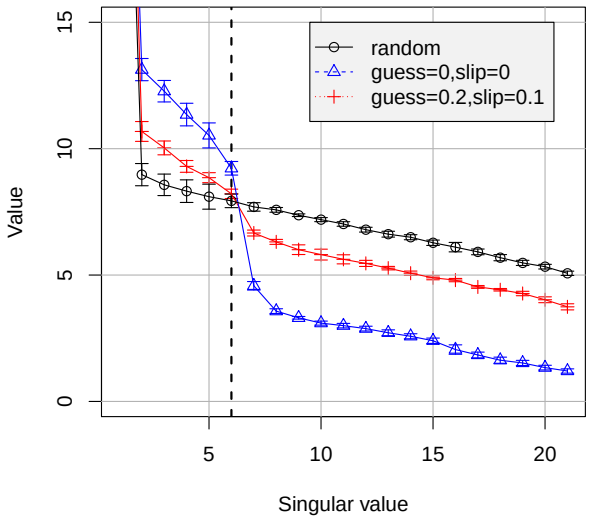



Figure 2: Singular values of simulated data for a 21 items test. Unit standard error bars for a 10-fold simulations is drawn for each line. A vertical dashed line at singular value 6 corresponds to the number of underlying latent skill factors.

### 2.2 Results

The results of the SVD method are shown in figure 2. The $x$ is the index of the singular value, and the $y$ axis is its actual value. Recall that the singular values of SVD indicate the strength of latent factors.

Three conditions are reported in figure 2. The $y$ values at 1 on the $x$ scale are truncated on the graph to allow a better view of the interesting region of the graph, but the highest value is from the [guess=0, slip=0] condition and the lowest is for the *random* condition. The random curve condition can be obtained by simulating random $\{0, 1\}$ values and ensuring that the overall average score of the results matrix reflects the original's data average. In this random condition, the slope from singular value 2 to 21 remains relatively constant, suggesting no specific number of skills. In condition [guess=0, slip=0], a sharp drop occurs between singular values of 6 and 7. Then the slope remains relatively constant from values 8 to 21. The largest drop is clearly at value 6 which corresponds to the underlying number of skills. In the third condition [guess=0.2, slip=0.1], the largest drop still remains visible between 6 and 7, but not as sharp as for the noiseless condition, as expected.

In other experiments with various number of skills, not reported here due to space constraints, we observed similar patterns. Another observation is that the random curve intersects with the other two after the number of underlying latent skills (after 6 in figure 2's experiment).

Therefore, the SVD method does allow for the identification of the number of skills with synthetic data, at least up to the [guess=0.2, slip=0.1] level.

## 3. WRAPPER-BASED METHOD

We introduce a second method to determine the number of dominant skills behind items based on a *wrapper* approach. In statistical learning, the *wrapper* approach refers to a general method for selecting the most effective set of variables by measuring the predictive performance of a model with each variables set (see [6]). In our context, we assess the predictive performance of linear models embedding different number of latent skills. The model that yields the best predictive performance is deemed to reflect the optimal number of skills.

### 3.1 A Linear Model of Skills Assessment

The wrapper method requires a model that will predict item outcome. A linear model of skills is defined for that purpose on the basis of the following product of matrices:

$$\mathbf{R} = \mathbf{QS} \tag{2}$$

where the $\mathbf{R}$ matrix contains observable student results with item rows and student columns, and the $\mathbf{S}$ matrix is the skills (rows) per students (columns) mastery profile (see for e.g., [3]). Matrix $\mathbf{Q}$ is the Q-matrix that maps items (rows) to skills (columns). Normalizing row sums of $\mathbf{Q}$ to 1 would yield values of 1 in the results matrix, $\mathbf{R}$, if all skills necessary to succeed an item is mastered by the corresponding individual. Equation (2) represents a *compensatory* interpretation of skills modeling, where each skill contributes additively to the success of an item.

A *conjunctive* model can be defined according to the following equation [1; 4] :

$$\neg\mathbf{R} = \mathbf{Q}\neg\mathbf{S} \tag{3}$$

where the operator $\neg$ is the Boolean negation, which is defined as a function that maps a value of 0 to 1 and any other value to 0. This equation will yield values of 0 in $\mathbf{R}$ whenever an examinee is missing one or more skills for a given item, and yield 1 whenever all necessary skills are mastered by an examinee.

### 3.2 Overview of the method

To estimate the optimal number of skills, the *wrapper* model can either correspond to equation (2) or (3). We will focus our explanations around equation (2), but they obviously apply to (3) if $\mathbf{R}$ and $\mathbf{S}$ are negated.

This model states that, given estimates of $\mathbf{Q}$ and $\mathbf{S}$, we can predict $\mathbf{R}$. We refer to these estimates as $\hat{\mathbf{Q}}$ and $\hat{\mathbf{S}}$, and to the predictions as $\hat{\mathbf{R}} = \hat{\mathbf{Q}}\hat{\mathbf{S}}$. The goal is therefore to derive estimates of $\hat{\mathbf{Q}}$ and $\hat{\mathbf{S}}$ with different number of skills and measure the residual difference between $\hat{\mathbf{R}}$ and $\mathbf{R}$.

First, $\hat{\mathbf{Q}}$ is learned from an independent set of training data. Then, $\hat{\mathbf{S}}$ is learned from the test data, and the residuals are computed[2].

[2]Note that computing $\hat{\mathbf{S}}$ from the test data raises the issue of over-fitting, which would keep the accuracy growing with the number of skills regardless of the "real" number of skills. However, this issue is mitigated by using independent learning data for $\hat{\mathbf{Q}}$, without which, we empirically observed, the results would deceive us: in our experiments using both $\hat{\mathbf{S}}$ and $\hat{\mathbf{Q}}$ from NMF while increasing the rank of the factorization (number of skills), ends up increasing prediction accuracy even after we reach beyond the "real" number of skills. This can reasonably be attributed to over-fitting.

An estimate of $\hat{\mathbf{Q}}$ is obtained through Non-negative Matrix Factorization (NMF). Details on applying this technique to the problem of deriving a Q-matrix from data is found in [3] and we limit our description to the basic principles and issues here.

NMF decomposes a matrix into two matrices composed solely of non-negative values. Its structure is equivalent to equation (2). The technique requires to choose a rank for the decomposition, which corresponds in our situation to the number of skills (i.e. number of columns of $\mathbf{Q}$ and number of rows of $\mathbf{S}$). Because NMF constrains $\mathbf{Q}$ and $\mathbf{S}$ to non-negative values, their respective interpretation as a Q-matrix and a as student skills assessments is much more natural than other matrix factorization techniques such as Principal Component Analysis, for example. However, multiple solutions exists to this factorization and there are many algorithms that can further constrain solutions, namely to force sparse matrices. Our experiment relies on the R package named NMF and the Brunet algorithm [5].

Once $\hat{\mathbf{Q}}$ is obtained, then the values of $\hat{\mathbf{S}}$ can be computed through linear regression. Starting with the overdetermined system of linear equations:

$$\mathbf{R} = \hat{\mathbf{Q}}\hat{\mathbf{S}} \tag{4}$$

which has the same form as the more familiar $\mathbf{y} = \mathbf{X}\beta$ (except that $\mathbf{y}$ and $\beta$ are generally vectors instead of matrices), it follows that the linear least squares estimate is given by:

$$\hat{\mathbf{S}} = (\hat{\mathbf{Q}}^T\hat{\mathbf{Q}})^{-1}\hat{\mathbf{Q}}^T\mathbf{R} \tag{5}$$

Equation (5) represents a linear regression solution which minimizes the residual errors ($||\mathbf{R} - \hat{\mathbf{Q}}\hat{\mathbf{S}}||$).

### 3.3 Prediction Accuracy and the Number of Skills

We would expect the model with the correct number of skills to perform the best, and models with fewer skills to underperform because they lack the correct number of latent skills to reflect the response patterns. Models with greater number of skills than required should match the performance of the correct number model, since they have more representative power than needed, but they run higher risk of overfitting the data and could therefore potentially show lower accuracy in a cross-validation. However, the skills matrix $\hat{\mathbf{S}}$ obtained through equation (5) on the test data could also result in over-fitting that will *increase* accuracy this time. We return to this issue in the discussion.

We use the same simulated data as described for the SVD method in section 2.1, where six skills are used to generate data according to the Q-matrix of figure 1. For this experiment, we only report the condition of guess=0.2 and slip=0.1.

Figure 3 shows the percentage of correct predictions of the models as a function of the number of skills. Given that predictions are $\{0, 1\}$, the percentage can be computed as $||\mathbf{R} - \hat{\mathbf{Q}}\hat{\mathbf{S}}||/mn$, where $m$ and $n$ are the number of rows and columns of $\mathbf{R}$.

The results confirm the conjectures above: the predictive accuracy increases until the underlying number of skills is reached, and it almost stabilizes thereafter. Over-fitting of $\hat{\mathbf{S}}$ with the test data is apparently not substantial.

It is interesting to note that the accuracy increments of figure 3 are relatively constant between each skill up to 6. This

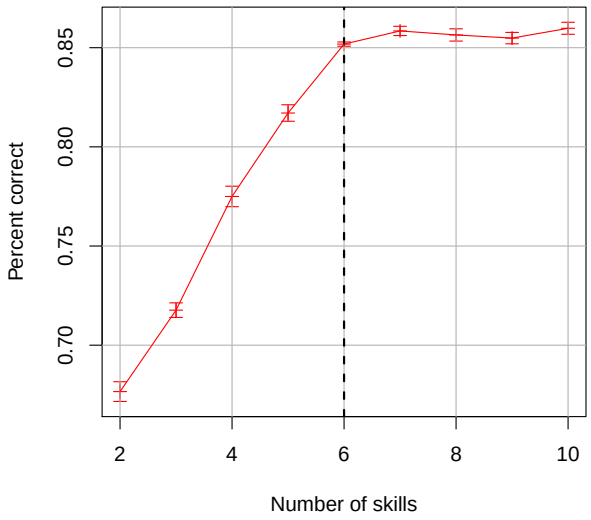



Figure 3: Precision of student results predictions from estimated skill matrix (equation (5)). Error bars are the standard error of the accuracy curves. Experiment is done with simulated data with 6 skills and *slip* and *guess* values of 0.1 and 0.2 respectively.

is also what we would expect since every skill in the underlying Q-matrix has an equivalent weight to all others. We expect that differences in increments indicate differences in the weights of the skills. This could either stem from the structure of the Q-matrix (for e.g., more items can depend on one skill than on another), or on the criticality of the skill over its item outcome.

# 4. APPLICATION OF THE METHODS ON REAL DATA FROM FRACTION ALGEBRA

Simulated data reveals that both the SVD and wrapper methods provide effective means to identify the number of latent skills. Are these means as effective in identifying skills with real data? This can depend on a number of factors. One factor is the degree to which a skill is determinant to the success of an item. General high level skills can only add to the chances of success, they are not decisive. More specific skills can be decisive, but there may be alternative skills that also account for an item success (e.g. a different method of solving a problem). Finally, noise from slips and guesses will undermine the ability of any method that attempts to identify the number of latent skills.

Therefore, an answer to the above question, i.e. whether we can identify the number of latent skills, is only valid within a given context, where the factors mentioned above take on a particular combination. So any conclusion will have to take into account this limitation in its generalization.

We investigate the question with data from Vomlel [10] on fraction algebra problems. This data set is composed of 20 question items and answers from 148 students. A Bayesian

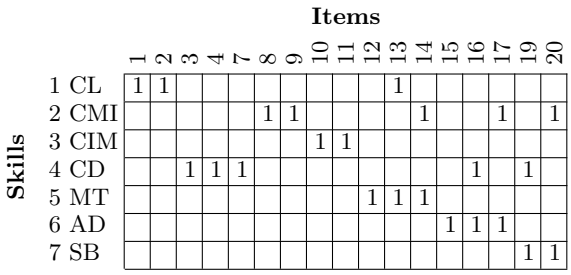



Figure 4: Conjunctive Q-matrix of Fraction Algebra data composed of 7 skills and 17 items. Item numbers refer to the original data items.

Network linking items to skills was defined by experts for the 20 items. It can readily be transformed into the Q-matrix shown in figure 4.

This Q-matrix is a subset of the whole Q-matrix from the Bayesian Network in Vomlel's study. It was chosen based on four fundamental skills of fraction algebra :

**1 CL:** cancelling out
**2 CIM:** conversion to mixed numbers
**3 CMI:** conversion to proper fractions
**4 CD:** finding common denominator

A total of 15 items are involved those skills. Because some items involved other skills, 3 more skills are added through conjunction, for a total of 7 skills:

**5 AD:** addition
**6 SB:** subtraction
**7 MT:** multiplication

And 2 more items involving these added skills are also added, for a total of 17 items. Six out of the 17 items involve a conjunction of 2 skills, whereas all other items are single skill.

Note that contrary to the synthetic data, skills are not expected to have equal weight in the prediction results, as some are only involved in two items, whereas others are involved in five items.

The SVD and wrapper methods are applied to the data in an attempt to derive the number of underlying skills. For the SVD method, the factorization is conducted on the full data set since this method does not rely on a cross validation process. For the wrapper method, the data is split in half for training, half for testing. Both approaches follow the methodology described in sections 2 and 3.

## 4.1 SVD method

Results of applying the SVD method to the fraction algebra data is reported in figure 5. Apart from the usual steep slope from singular value 1 to 2, there is no clear indication of the number of skills in this figure when we look at a change of slope as we had with the simulated data experiment. However, the *random* and *real* curves meet at singular value 2, which, according to the results from simulated data, would suggest that the number of latent skills is 2. However, this not consistent with the expert Q-matrix. It is also counter-intuitive since we would expect that more than two skills in fraction algebra problems would cover the skills described above.

We could also conclude that there is a continuum of skills, and/or that the data is too noisy to show any effect of skills.

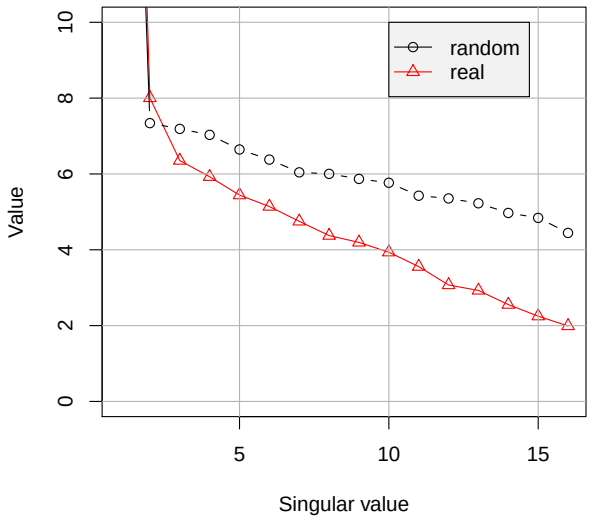



Figure 5: SVD results over fraction algebra data. The random and real curve at skill 1 are not shown but they are respectively 30 and 35.

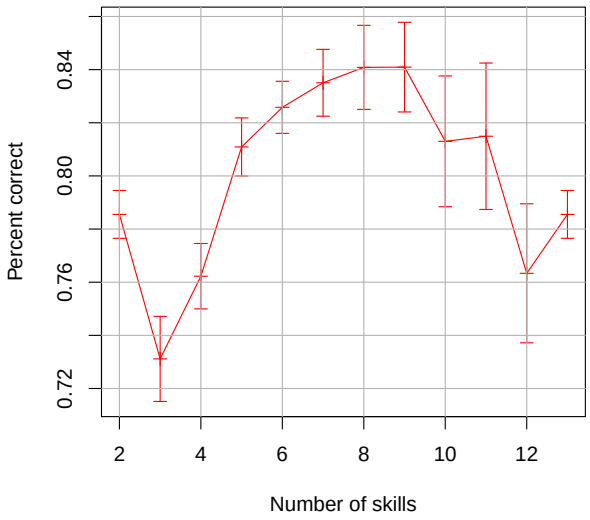



Figure 6: Wrapper method applied to the fraction algebra data set. The error bars represent the standard error of 50 folds results.

Let us turn to the wrapper method before speculating any further on these unexpected results.

## 4.2 Wrapper method

For the wrapper method, the data set is divided into two random samples of half the size of the original 148 students. One half is used for deriving the Q-matrix and the other in deriving the skills matrix, $\hat{\mathbf{S}}$, and measuring the accuracy of the predictions. This procedure is the same as the one used for the simulated data. As we explain below, a large number of folds (50) have to be run in order to obtain stable results.

Figure 6 reports the results of the wrapper method. We observe a sharp drop after skill 2, which suggests that a peak was reached at that point[3]. In that respect, it confirms the 2-skill findings of the SVD method.

However, we also observe a steady increase of accuracy starting from 3 skills, up to 8 skills, and a gradual decrease of skill contribution to performance starting from 4 skills. Except for the unexpected drop after 2 skills, this finding is close to the 7 skills defined by experts. And the fact that some skills have a greater weight on the performance is also consistent with the gradual decrease of contribution up to 8 skills.

Concerning the decrease after 9 skills, this can be explained by over-fittins in the NMF Q-matrix induction ($\hat{\mathbf{Q}}$) with the training data. In simulated data, the sample size was apparently large enough to shield the results from the over-fitting issue, but the smaller sample size of the real data may raise this issue here. Moreover, as the number of latent factors approaches the number of items in the data (17), the over-fitting issue becomes even more significant.

Drawing conclusions from this experiment with real data is obviously hard. Both the SVD and the wrapper methods seem to suggest that 2 skills would are plausible, but the wrapper method also points to an 8 skills set that is more consistent with the expert Q-matrix.

[3]The implementation of the method does not allow a computation of the accuracy for a single skill, but we can reasonably assume that a single skill model would perform worst than a 2-skills model.

# 5. DISCUSSION

Both the SVD and the wrapper methods provide strong cues of the number of underlying skills with simulated student test data. However, for the Vomlel data set, both methods yield results that are much more ambiguous. Instead of the 7 skills that were identified by experts over the 17 items set, the SVD method suggests only 2 skills if we rely on the intersection with the random data curve, and no clear number if we look for a change of slope after skill 2. The wrapper method shows data that is also consistent with 2 skills to the extent that a drop of accuracy is observed at 3 skills, but a rise of accuracy up to 8 skill draws an interpretation closer to the experts' 7 skills set.

An important difference between the SVD and the wrapper methods has to do with the independence of skills. For SVD, orthogonality of the singular matrices $\mathbf{U}$ and $\mathbf{V}$ in equation (1) forces latent factors to be independent. NMF does not require latent factors to be independent. The orthogonality constraint of may limit the application of the SVD method with respect to real skills and might explain some of the difference between the two methods. The skills from the synthetic data of the first experiment were independent and the Q-matrix had an homogeneous pattern for each skill, and therefore the effect of dependence between skills could not come into play.

Obviously, the study calls for more investigations. The findings from one set of data from the real world may be

highly different from another set. More studies should be conducted to assess the generality of the findings. Other investigations are called for to find ways to improve these methods and to better understand their limits when faced with real data. In particular, we need to know at which level of noise from guess and slip factors do the methods break down, and what is the ratio of latent skills to data set size that is critical to avoid over-fitting of the wrapper method. One improvement that can be brought to the wrapper method is to use a cross validation to derive the skills matrix. This would require the use of two sets of items, one for testing and one for assessing the student's skills. This comes at the cost of a greater number of items, but it avoids the problem of over-fitting that leads to accuracy increases.

## 6. ACKNOWLEDGEMENTS

This project was supported by funding from the MATI institute (www.matimtl.org) and by Canada's NSERC discovery program.

## 7. REFERENCES

[1] T. Barnes. The Q-matrix method: Mining student response data for knowledge. In *AAAI'2005 Workshop on Educational Data Mining*, pages technical report WS–05–02, 2005.

[2] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1995.

[3] M. C. Desmarais. Conditions for effectively deriving a q-matrix from data with non-negative matrix factorization. In C. Conati, S. Ventura, T. Calders, and M. Pechenizkiy, editors, *4th International Conference on Educational Data Mining, EDM 2011*, pages 41–50, Eindhoven, Netherlands, June 6–8 2011.

[4] M. C. Desmarais, B. Beheshti, and R. Naceur. Item to skills mapping: deriving a conjunctive q-matrix from data. In *11th Conference on Intelligent Tutoring Systems, ITS 2012, Chania, Greece, 14–18 June 2012*.

[5] R. Gaujoux and C. Seoighe. A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, 11(367):`http://www.biomedcentral.com/1471--2105/11/367`, 2010.

[6] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[7] A. Mitrovic, M. Mayo, P. Suraweera, and B. Martin. Constraint-based tutors: A success story. In L. Monostori, J. Váncza, and M. Ali, editors, *IEA/AIE*, volume 2070 of *Lecture Notes in Computer Science*, pages 931–940. Springer, 2001.

[8] P. I. Pavlik, H. Cen, and K. R. Koedinger. Performance factors analysis –a new alternative to knowledge tracing. In *Proceeding of the 2009 conference on Artificial Intelligence in Education*, pages 531–538, Amsterdam, The Netherlands, 2009. IOS Press.

[9] P. Suraweera, A. Mitrovic, and B. Martin. A knowledge acquisition system for constraint-based intelligent tutoring systems. In C. K. Looi, G. McCalla, B. Bredeweg, and J. Breuker, editors, *Proceedings of the 12th International Conference on Artificial Intelligence in Education, AEID'2005*, pages 638–645, Amsterdam, July 18–22 2005. IOS Press.

[10] J. Vomlel. Bayesian networks in educational testing. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 12:83–100, 2004.

[11] T. Winters, C. R. Shelton, and T. Payne. Investigating generative factors of score matrices. In R. Luckin, K. R. Koedinger, and J. E. Greer, editors, *Artificial Intelligence in Education, Building Technology Rich Learning Contexts That Work, Proceedings of the 13th International Conference on Artificial Intelligence in Education, AIED 2007, July 9-13, 2007, Los Angeles, California, USA*, volume 158 of *Frontiers in Artificial Intelligence and Applications*, pages 479–486. IOS Press, 2007.

# Mining Student Behavior Patterns in Reading Comprehension Tasks

Terry Peckham
Department of Computer Science
University of Saskatchewan
110 Science Place
Saskatoon, SK., Canada
tep578@mail.usask.ca

Gord McCalla
Department of Computer Science
University of Saskatchewan
110 Science Place
Saskatoon, SK., Canada
mccalla@cs.usask.ca

## ABSTRACT

Reading comprehension is critical in life-long learning as well as in the workplace. In this paper, we describe how multidimensional k-means clustering combined with Bloom's Taxonomy can be used to determine positive and negative cognitive skill sets with respect to reading comprehension tasks. This information could be used to inform environments that support students improving their meta-cognitive skills.

## Keywords

Data mining, k-means clustering, Bloom's taxonomy, reading comprehension, cognitive strategies.

## 1. INTRODUCTION

Anderson and Pearson [4] in their seminal work on reading comprehension describe three different cases where reading comprehension is a problem. First, a person having difficulty reading is likely to have gaps in knowledge. Prior knowledge is necessary in the determination of what he/she can currently comprehend. Second, the reader can have an incomplete understanding of the relationships that exist among facts on a certain topic. Since the current knowledge base is used to create all of the relationships on a topic, new arbitrary information can be a source of confusion, slow learning and slow processing, which leads to unsatisfactory reasoning. Third, readers are unlikely to be able to make correct inferences about the material in order to arrive at a coherent overall representation of the topic. The creation of a coherent representation for a topic requires the drawing of precise, integrated inferences. Often poor readers do not perform these tasks either routinely or spontaneously [8]. Any reading comprehension tools or models need to be able to address these problems with deep comprehension.

The reading strategy instruction method is one of the most often suggested methods for enhancing reading ability [18, 20]. This particular method deals with problems on the vocabulary and sentence levels [2], and on higher level issues such as text comprehension [2,14]. Other recommended approaches include determining the main message of the content (e.g. summarization), the use of textual enhancements (e.g. illustrations, mental images), question and answer drills (e.g. self-questioning) and practicing meta-cognition (e.g. through comprehension monitoring) [9]. However, the most successful reading strategies combine methods rather than one single technique [14].

There are several barriers to the adoption of multiple strategies within the classroom setting [14]. First, there is a large amount of training that is required for the teachers to become familiar with the strategies in order to employ them within the classroom setting. Second, there is a considerable time requirement for teachers to prepare the course materials. Third, getting the students to apply the strategies in daily life can be extremely complex. Therefore, the creation of environments that help relieve the teacher of some of these complexities would be of great benefit.

There are several learning environments that aid students with their reading comprehension. Some of the more prominent are Project Listen, iSTART, Point&Query, and AutoTutor. Project Listen [13] creates an environment where children and ESL (English as a second language) students can read text out loud with the aim to improve this skill. The software listens to the reader and makes suggestions on how to improve their reading skills. One of the ways that the software increases reading comprehension is by asking the students questions about the text that they just read [6]. Presumably, the increase in reading comprehension and word comprehension do not translate into helping the students enhance the deeper comprehension skills discussed by Anderson and Pearson [4] since this is not the aim of this particular software. The remaining environments, however, do take aim at creating deeper understanding within the reading comprehension field. Point&Query augments current learning environments, such as hypertext and hypermedia, by providing learner controlled question and answer sessions that expose readers to deep causal questions [10]. Both AutoTutor and iSTART make use of animated agents and natural language dialogue to scaffold inquiry strategies, metacognition, and explanation construction [10]. AutoTutor generates why, what-if, and how style questions and then enters into a dialogue with the student to expose the deeper constructs of the topic. iSTART takes a coaching approach to teach the students how to construct and improve self-explanations combined with other metacomprehension strategies. Although these systems have demonstrated student learning gains and improvement in learning strategies, more can still be done.

Many of the aforementioned tools, created to aid in reading comprehension, are more closed-ended systems that require a significant amount of time and energy to develop course content [15]. These closed-ended systems often make use of help requests to aid them in determining when a student is having a reading comprehension problem [6,10,13]. However, the vast

majority of environments such as WebCT/Blackboard, Moodle, etc., that are adopted by schools and post-secondary institutions are more open-ended in nature. These systems provide much more flexibility in terms of content development and improved ease in making changes to the content compared to what can be provided with closed ended systems. The problem with the open-ended systems is that they provide no real support for student learning other than providing the content for the students. However, open-ended systems do have good tracking facilities in place to capture student interaction with the system. By making use of current data mining techniques and pedagogy aimed at improving student learning, it is possible to capture students' cognitive behavior from these open environments.

Trace methodologies, such as capturing keystroke data, events, eye tracking data, etc., have demonstrated that data generated from a student's interaction with an environment can provide the necessary information to make cognitive and metacognitive interpretations [5]. This makes sense since how a student consumes content will have a direct effect on their comprehension of that content. If we know the following: what task the student is currently working on, the difficulty of the task, and the current behavior of the student as they work on the task, we can make cognitive interpretations [5,13,17,21]. Bloom's Taxonomy [3] of the Cognitive Domain, provides a pedagogical framework for determining how cognitively difficult a question/task is. Using this framework we can determine if the student's current cognitive skills are appropriate for the task that they are currently working on.

Bloom's Taxonomy [7] and its subsequent revision [3] are comprised of three overlapping domains: cognitive, affective and psychomotor. The affective domain is comprised of emotions, attitudes and values. The psychomotor domain is comprised of physical skill mastery, coordination, etc. The cognitive domain provides a method to classify educational objectives that relate to knowledge [21]. Within the cognitive domain are six hierarchical levels in order of increasing complexity. They are: knowledge, comprehension, application, analysis, synthesis and evaluation (as revised by Anderson et. al.[3]). The first three levels are considered to be foundational learning and are based upon the ability to know and apply factual knowledge [21]. The last three levels are considered higher level learning that is more abstract in nature [20]. Bloom had originally assumed that you could not achieve the higher levels without first mastering the lower levels of the hierarchy [7]. However, it appears that it is possible to work at the higher levels on some topics without first mastering the lower levels [3].

Wankat and Oreovicz [19] provide some examples of how to apply Bloom's taxonomy to an engineering domain. Knowledge or recall involves the descriptions, definitions, generalizations and other routine information about a topic. Comprehension involves understanding the technical representations of a topic including the translation, interpretation and extrapolation of that topic. Application involves the use of topical abstractions in explicit situations such as the use of rules, procedures and theories to perform some computation. Analysis involves breaking a problem into its principal parts in order to highlight any content hierarchy, properties. Furthermore, connections and structure found within the content are defined and clarified. Synthesis involves putting together all the constituent parts of a problem into a coherent system or solution. This can be very difficult since the process is open-ended and there may be many possible solutions to the problem. Lastly, evaluation can involve making conclusions about the value of materials used in a project or the methods used in that project. There is a need to satisfy specific criteria or use some standard of appraisal.

Through the use of the different levels of Bloom's Taxonomy and questions that are appropriately couched within the framework, it is possible to help learners to overcome the various problems originally posed by Anderson and Pearson [4].

## 2. METHODOLOGY

An experiment was designed to look for patterns of student behavior in a reading comprehension task. Students interacted with a learning environment designed to emulate hypermedia courses offered in post-secondary institutions where written content is presented along with questions about that content. The students could view the content and/or questions in any order or manner they chose with no constraints applied to their interaction with the system. In keeping with trace methodology approaches, all of the interactions/events with the content and questions were recorded and time-stamped. These would include events such as mouse click, mouse wheel, which item was clicked or selected and so on.

To aid in determining what part of the document was currently being read, a small scrollable text box that allowed 7 lines of text to be displayed as displayed in Figure 1. The size of the text box performed a couple of tasks. First, it does not take more than one minute to read the approximately 77 words contained within the text box. Although not directly used in the analysis, this could be used to determine if the individual was distracted from the task at hand. Second, it provided a means to determine how much time and how quickly the student read over the portions of the document that contained the answers to the various questions. The questions could be selected in any order and any text the students had entered into the answer text box was saved and displayed when the corresponding question was selected. None of the participants was observed, nor reported, as having any difficulty with operating the interface.

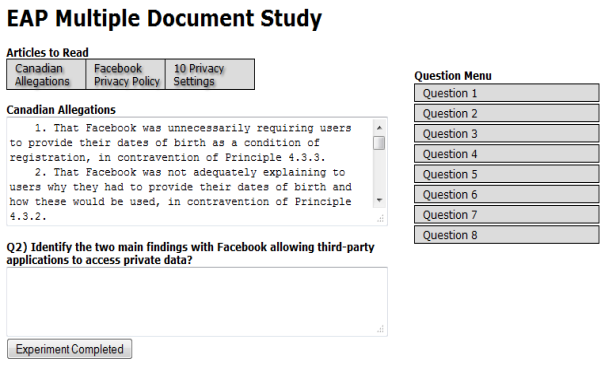



**Figure 1 Screen Capture of Interface**

The questions were developed using Bloom's Taxonomy Action Verbs [3,7]. Bloom and Anderson created a list of verbs that direct the way that a question should be answered. These verbs correspond to a level within Bloom's taxonomy. When you place the action verb at the beginning of the question, it frames the way that the question should be answered [7]. For example, Bloom's lowest level, knowledge, contains the action verb 'list'. Since the

task of the knowledge level is to remember previously learned information, successfully listing something that the student has previously read would demonstrate that the student has mastered that level of cognitive difficulty for that content. Questions at various Bloom levels were presented to the participants in a randomized order. All of the questions were present on the screen at all times and could be selected in any order by the student.

Questions were scored in order to provide a metric for how well the students were comprehending the content. In order to deal with the subjective nature of scoring question answers, a rubric was created according to the principles laid out in [16] for each of the questions. The rubric was revised a couple of times to take into account the various types of answers that were submitted during the testing phase of the development. For the lower Bloom levels, the answers generally came from one direct location within a document and so the scoring was fairly simple. For the higher level Bloom questions, information from multiple sources was expected. It was also expected that the students would bring their own prior knowledge to bear on the answer. It was here that revisions were required as the beta testing group interpreted the questions in unforeseen ways. The experiment was broken into two components. The first component provided the students with a single document to read and questions about that content were based upon the lower levels of Bloom's Taxonomy. All of the answers to the questions could be found within the document. The document that was chosen was a fairly technical document based upon Canadian privacy law as it applies to Facebook so that the participants would not have much prior knowledge of the specific subject matter. The participants were given 30 minutes to finish reading the document and answering four questions.

The second condition provided the students with two more documents in addition to the first document. The purpose of this condition was to better test the higher levels of Bloom's Taxonomy. The higher levels of Bloom's Taxonomy require synthesis and evaluation and so more information and documents were needed to allow for these requirements. The second document was instruction on how to implement advanced privacy features not commonly used within Facebook and the third document was a high level overview of the privacy settings used within Facebook. Again the answers to the high level questions could be found within the documents provided. However, in order to fully answer the higher level questions, information from more than one document was required. For this second condition, 90 minutes were allotted as the questions were more difficult and there were two new documents that needed to be read to generate complete answers. Two questions were aimed at the prior reading done in the first condition. One was a repeated question from the first condition and a second question was new but based solely on the information found in that first document. The remaining six questions were new and tested various levels of Bloom's Taxonomy. It was possible that the students could answer the questions in an increasing level of difficulty; however, they would have to purposefully select that order since the order in which they were presented was random.

Since the amount of time required to participate in both conditions might be a factor in participant involvement, both conditions were designed so that they could be run separately and using different participants depending on the participants' wishes. In the actual running of the experiment, the majority of the participants moved from the first condition right into the second condition with no delay. The participants were adult students enrolled in a grade 12 Saskatchewan Institute of Applied Science and Technology (SIAST) Adult Education English course. There were 17 participants for the first condition and 11 for the second condition with an average age of 26.

## 3. RESULTS

The 28 participants generated over 8500 events in total from both conditions. Events such as the mouse clicking on a specific button or object and mouse wheel scrolling were captured. Each event was time-stamped with the user-id, event-id, current question-id, current document-id, and position within the current document. This gave us what task/question the student was currently working on, which document they were working on, where in the document they were, and what event they were using. For example, if the student moved the scroll wheel of the mouse to move down in the document we could then determine from the time-stamp data and the position data, how quickly and what material they were reading. With this information we can begin to deduce student behavior as they work at completing the various questions.

In order to determine how much reading the students were doing, the timestamp data was processed so that reading, scanning and scrolling navigation times could be calculated for each interaction/event. The time cutoffs used to distinguish reading from scanning from scrolling fit with other document navigation research [1]. Any time between events greater than five seconds was classified as reading. Any time greater than two seconds but less than five seconds was classified as scanning and any time less than two seconds was classified as scrolling. The reading time also encompassed time that the participant spent thinking about the answer. In the 8500 events captured across the 28 participants, only 13 events had a time greater than two minutes and only 33 events had a time greater than one minute before another event was performed. Given the time it takes to read the content in the textbox, the total time between events including the reading and thinking times, was not a large enough percentage of the data to warrant separate classification.

The total amount of time that a student spent in the experiment was calculated and used to create a ratio of time spent by the student reading, scanning and scrolling. This ratio was then broken down into the reading, scanning and scrolling ratios for each individual question. When combined with the level of difficulty for each question, as determined by Bloom's taxonomy, it was possible to tie student reading behavior to the difficulty of the task.

In order to see if there were students who behaved similarly for different levels of difficulty, we implemented the Forgy method for k-means clustering for d=3 dimensions and k=4 [11]. Hammerly et al. [11] demonstrated that the Forgy method for initialization was the preferred method for initializing the standard k-means, also known as Llyod's, algorithm [12]. The dimensions that we chose were the reading, scanning, and scrolling axes. K = 4 was chosen since our sample size was small. More than 4 clusters produced some clusters where there were too few to be statistically analyzed. Since the algorithm randomly chooses its centroid points, there is no researcher bias entering into the initial sets of clusters that were created. In order to find as many interesting clusters as we could, the Forgy k-means algorithm was iterated multiple times. We defined interesting clusters as those clusters that elicited either positive or negative reading, scanning or scrolling behaviors. A positive behavior is defined as a

behavior that results in a good grade. A negative behavior is defined as a behavior that results in a poor grade. Those clusters that presented with both positive and negative behaviors were deemed less interesting. Each time an interesting cluster was found, the centroid was recorded. Once multiple interesting centroids were found, the most interesting centroid found was hard coded as a starting centroid. The hard coding of the algorithm removes one of the random initializations from the Forgy initialization and inserts the most interesting centroid in its stead. For example, the experiment used k = 4 random clusters in the initialization. With the hard coded cluster added, k = 3 random and k = 1 hard coded are what the algorithm would initialize with. The algorithm was run again with one hard coded centroid and three randomly chosen centroids to see how the other random clusters interacted since how the cluster is initialized is known to have an effect on how the other clusters form [11]. If a new interesting cluster was discovered that was more predictive of students' behavior than a previous closely related centroid, the old centroid was removed in favor of the new centroid. If no more centroids were discovered that were more interesting than the hard coded centroid, then the second most interesting centroid was hard coded and the remaining two centroids were left random and the above process was duplicated with two hard coded centroids. A third hard coded cluster was added in accordance with the above procedure and the process was performed again until all four of the initializations centroids were hard coded.

Over multiple iterations six interesting clusters were discovered with two of these clusters containing too few data points to be included in any statistical analysis that was performed. The following clusters proved to be statistically interesting with respect to the Bloom level:

- Light Reading Cluster: 50% reading: 30% scanning: 20% scrolling (50:30:20)
- Light Medium Reading Cluster: (60:30:10)
- Heavy Medium Reading Cluster: (70:20:10)
- Heavy Reading Cluster: (80:10:10)

The two clusters, Medium Scrolling (10:10:60) and Medium Scanning (10:60:10), were clusters that we expect to play a more important role in future experiments. However, due to our sample size, they could not be used in our statistical analysis.

An ANOVA was performed on each of the clusters to see if a statistically significant relationship could be found between the different reading behaviors as clustered by k-means and the Bloom levels of the questions within the experiment. The tests were performed at the $\alpha = 0.05$ level. Questions at Bloom levels 1,2,3,5, and 6 were provided in this experiment. There were no Bloom level 4 questions, to give learners time to answer more questions at level 5 and 6 within the overall time constraints.

Table 1 shows that, with the exception of level 5, all of the Bloom levels were statistically significant. The null hypothesis used for these tests are that the means for each of the clusters does not vary according to the Bloom level that is being tested. In other words, the reading, scanning and scrolling means should be the same for all of the clusters found by k-means. Table 1 shows that the differences found between the clusters for each of the Bloom levels were not due to random chance. The p-values indicate that, in all but two cases, there is a really small chance of getting these results if no real difference between the groups exists. This indicates that the students' reading, scanning and scrolling behaviors captured by the system and then clustered are significantly different from one another as it relates to the level of Bloom's taxonomy. For example, those students who were classified as Light Readers based on the reading, scanning and scrolling ratios for Bloom level 1 were significantly different from those who were classified as Light Medium Readers for the same Bloom level. However, the ANOVA itself cannot make this exact determination of which cluster is significantly different from another cluster; it can only tell us that there is a significant difference between some of the groups in the analysis. Further analysis, discussed later on, is needed in order to see which of the clusters are significantly different from each other.

Although inclusion in a cluster does not completely predict scores, it is indicative of overall performance. For example, take question 2 in the first condition (low level Bloom with a single document) that was designed to force the students to scan through the document as they needed to count the number of instances that a certain event, such as a successful appeal on a complaint about Facebook to the Canadian Privacy Commission, occurred in the document. This type of problem is often present in many forms in academia and the work place where it is necessary to arrive at a solution within the time constraints. 100% of the students in the Light Reading (50:30:20) cluster, which was higher in scanning and scrolling times, achieved full marks or close to full marks. Correspondingly, those students in the Heavy Reading (80:10:10) cluster scored no better than 50% with over ½ of the students in the cluster scoring 0%. Since the source materials were present for the duration of the experiment and there were time constraints, the Heavy Reading strategy is not the best strategy to be used in this situation. This result is somewhat surprising since it is generally accepted that Heavy Reading is considered a good cognitive strategy in a reading comprehension task. In this case, the cognitive load required to be able to answer this type of question, the time limitations of the experiment and the fact that the source materials were available, make the adoption of the Light Reading strategy a better choice. The reduction in the cognitive load by choosing to perform more scanning and scrolling through the document rather than committing the information to memory when performing Heavy Reading allows the participants to perform better on this type of task. It should be noted that for other tasks, a Heavy Reading strategy is the best choice. Furthermore, in situations where the source materials are not available during the task, the Heavy Reading strategy is most likely the best choice regardless of the task given.

The Heavy Reading strategy proved to be the most successful strategy as the level of difficulty for the questions increased as measured by Bloom's Taxonomy. The participants were able to achieve better marks compared to those that chose a Light Medium Reading strategy. For example, question six of the second condition required the participants to put together various thoughts and ideas about FaceBook privacy policy from multiple documents into a complete whole thought that did not exist in any of documents (Bloom level 6). For this problem the students fell into multiple clusters. Each document had its own set of events that tied the reading, scanning and scrolling ratios to that document. This provides a mapping of how each student used each document to answer a particular question. In order for the students to get a good grade they needed to fall into the Heavy Reading category on all the documents that were required to fully answer the question. Those students that performed Heavy Reading on all the necessary documents scored well. The students that performed Heavy Reading on only one of documents

they were required to read did not score above 30%. Those students that performed Heavy Reading on two of the required documents scored no higher than 83% and those that performed Heavy Reading on all of the documents scored no lower than 83% and up to 100%. Those students that used the Light Medium Reading strategy scored 0%. There was one student who scored 30% that used the Light Medium Reading strategy but their answer contained no content from any of the documents, rather they used extraneous information from their previous experience.

The use of the Light Reading strategy did not appear above Bloom level three and the Light Medium Reading strategy appeared throughout the Bloom levels. At Bloom level's five and six, those participants that chose to use the Light Medium Reading strategy did not receive good grades. The availability of the source documents to the participants did not aid them in answering more challenging questions. The participants needed to be able to recall information from a variety of sources in order to be able to fully answer the questions. Instead of using source material, probably because they could not recall where it was or if it was present, they used incorrect information from some other source outside of the experiment. It should be noted that they did not access supplementary material from either books or the Internet during this experiment.

These aforementioned patterns of behavioral clustering being predictive of marks do not hold in all cases. For example, the Light Reading (50:30:20) cluster for question 3 in the second condition (higher level Bloom with multiple documents) had 50% of the students achieve 100% while the other 50% received 0%. Since we captured the current reading position within the document with each event, we can determine the amount of time spent reading, scanning and scrolling over the portions of the document that contain the answer. When analyzed, this information is able to fully account for the differences in scores found within the cluster of the above example. For example, those students who received 0% spent the majority of their time scrolling and scanning compared to those who received full scores, who spent much more time reading over the portion of the document that contained the answer.

The Low Level and High Level analysis from Table 1 shows that when Bloom is broken down into two categories, low level (levels 1, 2, and 3) and high level (levels 5 and 6) there are significant differences for both the high and low levels between the clusters. When we perform the Tukey-Kramer test later on (Table 5), it will show that all of the clusters are significantly different from one another as well. Interestingly, when we combine all the levels together to see if the clusters by themselves are statistically different, we get no significant results. In other words, higher level and lower level meta-cognitive reading strategies seem to elicit different behavior on the part of learners.

In order to find out which clusters were significant from each other, a Tukey-Kramer analysis is required. A Tukey-Kramer analysis allows a pairwise comparison of each of the clusters and allows a comparison of groups that do not have the same number of students. The minimum significant difference value was used to calculate if the pairwise comparison was significant and correct for the multiple comparisons. The numbers in the top right hand portion of the Tables 2 through 5 show the Tukey-Kramer minimum significant differences (MSD). The numbers in the lower left corner of Tables 2 through 5 show the observed absolute value of the difference in means between each pair of groups. Those numbers in the lower left of the tables marked with an asterisk are deemed significant if they are larger than their corresponding MSD located in the top right of the table. Table 2 shows that all of the clusters were significantly different from each other. This was found for all of the other Bloom levels except for Bloom level 2 and 5. Table 3 shows that there are significant differences between most of the groups except for the Medium Heavy Reading cluster and the Heavy Reading cluster for Bloom level 2. Although the k-means algorithm clustered these reading, scrolling and scanning ratios into two different clusters, the actual differences between the ratios was close. So the grades tended to be higher in the Medium Heavy Reading and at the same time lower in the Heavy Reading cluster. It was situations like this one where the ratios were close together that made us wonder if a breakdown of individual Bloom levels was the best predictor or if the levels should be more coarse-grained and moved into a high level Bloom category and a low level Bloom category rather than individual Bloom levels.

One of the major problems with this experiment was that we did not have a large enough sample size for the higher levels of Bloom as tested in the second condition. Table 4 shows that there were no significant differences found between any of the clusters at Bloom level 5. A more in-depth analysis showed that most of the students chose a similar strategy to answer those questions and

| Bloom Level | F | P | F-Critical |
|---|---|---|---|
| 1 | 79.94 | 3.14E-16 | 2.86 |
| 2 | 39.31 | 3.74E-11 | 2.88 |
| 3 | 147.93 | 4.80E-11 | 3.63 |
| 5 | 0.60 | 0.63 | 3.59 |
| 6 | 50.77 | 0.000385 | 5.99 |
| Low Level | 209.48 | 1.83E-43 | 2.48 |
| High Level | 95.95 | 1.64E-17 | 2.86 |
| All Levels Combined | 1.40 | 0.25 | 2.68 |

**Table 1 One way ANOVA for Bloom Level**

| | **50,30,20** | **60,30,10** | **70,20,10** | **80,10,10** |
|---|---|---|---|---|
| **50,30,20** | - | 0.08311 | 0.07745 | 0.08089 |
| **60,30,10** | 0.16204* | - | 0.07976 | 0.08311 |
| **70,20,10** | 0.2963* | 0.13426* | - | 0.07745 |
| **80,10,10** | 0.4447* | 0.28268* | 0.14842* | - |

**Table 2 Tukey-Kramer Analysis Bloom Level 1 (* denotes significant differences)**

| | **50,30,20** | **60,30,10** | **70,20,10** | **80,10,10** |
|---|---|---|---|---|
| **50,30,20** | - | 0.21238 | 0.19337 | 0.17629 |
| **60,30,10** | 0.21341* | - | 0.2324 | 0.21839 |
| **70,20,10** | 0.4906* | 0.2772* | - | 0.19995 |
| **80,10,10** | 0.6724* | 0.459* | 0.18183 | - |

**Table 3 Tukey-Kramer Analysis Bloom Level 2**

| | 60,30,10 | 70,20,10 | 80,10,10 |
|---|---|---|---|
| **60,30,10** | - | 40.5993 | 28.4136 |
| **70,20,10** | 12.182 | - | 38.9069 |
| **80,10,10** | 11.666 | 0.5159 | - |

**Table 4 Tukey-Kramer Analysis Bloom Level 5**

| | 50,30,20 | 60,30,10 | 70,20,10 | 80,10,10 |
|---|---|---|---|---|
| **50,30,20** | - | 0.09225 | 0.07145 | 0.0668 |
| **60,30,10** | 0.15755* | - | 0.09473 | 0.09127 |
| **70,20,10** | 0.27206* | 0.11451* | - | 0.07019 |
| **80,10,10** | 0.4222* | 0.26463* | 0.15012* | - |

**Table 5 Tukey-Kramer Analysis Low Level Bloom**

therefore no significant differences were found between the clusters. With a larger sample size we believe that even this would become statistically significant. The fact that a significant difference was found in Bloom level 6 may just be due to an artifact in the data; however, the significant differences found in the lower Bloom levels 1 through 3, given the slightly larger N, seems to imply that with a larger N we will see those same differences in the higher Bloom levels.

For the low level and high level Bloom groupings, significant differences were found between all of the clusters. Table 5 shows the significant differences found for the low level Bloom grouping.

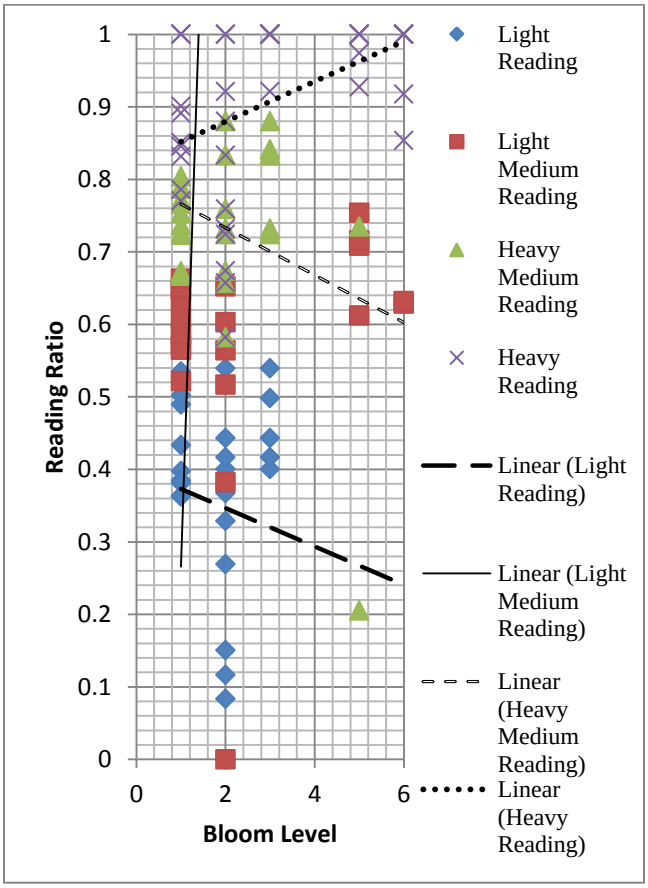



**Figure 2 Graph of Reading Ratio vs Bloom Level**

Next we analyzed how the clusters were related to the Bloom levels. Figure 2 shows that the Light Reading behavior was not found in any questions above Bloom level 3. This seems to indicate that Light Reading behavior is not conducive to the more cognitively difficult tasks. The Heavy Medium Reading cluster had only 2 instances in questions above Bloom level 3. The decreasing use of Heavy Medium Reading as the Bloom level of difficulty increases shows that some of the students' adapted a heavier reading behavior compared to the Light Reading behavior at the lower Bloom levels. They gave up the Heavy Medium Reading strategy for the Heavy Reading strategy used more in the higher Bloom levels. The Heavy Reading cluster was found at each of the Bloom levels. As the Bloom levels increase in difficulty, the amount of Heavy Reading increases until all but one student are Heavy reading at Bloom level 6. Correspondingly, the Light Reading cluster that contains more scrolling and scanning decreased as the Bloom level increased. This seems to suggest that different strategies are appropriate for different Bloom levels. There were several students that used the same cognitive strategy throughout the experiment despite the difficulty of the tasks changing. For example, some of the students chose a Heavy Reading strategy for the entire experiment. As a result they did not complete the experiment with respect to answering all of the questions as they spent too much time reading and not enough time answering the questions. Furthermore, students who chose a Heavy Reading strategy for the lower level Bloom questions did not always score very well even though the questions were cognitively simpler. The question 2 example from the first condition cited earlier in the paper is a good example. Other students chose different strategies for different levels of difficulty. For example, they would choose a strategy that was higher in scanning for the lower levels of Bloom and switch to a Heavy Reading strategy at the higher levels of Bloom. These students were able to complete the experiment and answer all of the questions within the time allotted.

At Bloom level 6 only two strategies are used: the (60:30:10) Light Medium Reading and the (80:10:10) Heavy Reading strategies. Although the students' inclusion in the Heavy Reading cluster was a good indicator of higher scores, there was still a lot of variance in the grades found within the Heavy Reading cluster for Bloom level 6. The best predictor of scores within the cluster was the ratio of reading time spent over the position in the various documents that contained the material necessary for the answers. This helped to identify those students that merely used their own unsupported opinions to answer questions versus those students that used information from the articles to support their answer.

## 4. CONCLUSIONS

This experiment demonstrates that the various cognitive strategies used by students to solve tasks of varying degrees of difficulty can be recognized automatically by an ITS. The use of Bloom's Taxonomy for categorizing the difficulty of the task and k-means clustering on the reading-scanning-scrolling strategies allowed for the detection of these cognitive strategies. These clusters can easily be turned into metrics that can be used by a system to discover the strategies the students are using and provide the necessary metacognitive suggestions to improve the student's cognitive skill set. Furthermore, the experiment shows that students may not always select the best strategy to use. This approach is not refined enough to predict an actual score on a question. However, it does provide a method of determining the reading strategy being used and predicting if the cognitive

strategy that is being employed is one that is positive or negative given the difficulty level (in terms of Bloom) of the question. Furthermore, since we are able to detect these inconsistencies in the use of cognitive strategies automatically we have the potential to automatically update a student model, and thereby inform the student about the metacognitive strategies they are employing and/or suggest appropriate pedagogical tasks that could be useful for a student attempting to improve weak metacognitive skills in the reading comprehension domain at least.

It is possible that the course grained detection of cognitive strategies will provide direction for systems where the application of more fine grained searches and algorithms might be able to predict the grade or allow for some specific pedagogical interventions. For example, do the students perform some course grained strategy in their initial search through a document and then use that information to refine their strategy for one that is more optimal for the solving some particular task?

K-means clustering comes with its benefits and drawbacks. The benefit of this algorithm is that it arrived at four interesting centroids that are hard-coded and that can be used in a real-time algorithm for the detection of significant reading strategies. There are other clustering methods, such as EM clustering, that may work better at determining new cluster centroids or are better at including the students in the correct cluster. This will be a subject of further research.

Future experiments also need to be performed to increase the sample size of the experiment, especially in terms of the higher Bloom levels. The increased sample size should allow us to see statistically significant cognitive skill differentiation at the higher Bloom levels but should also help to validate the reading-scanning-scrolling clusters that were not statistically viable with the current sample size. These experiments should further help solidify the use of Bloom's Taxonomy as a tool in detecting cognitive strategies for reading comprehension tasks. Furthermore, the interplay between reading comprehension and document selection may provide some interesting insights at the higher levels of Bloom's Taxonomy.

## 5. ACKNOWLEDGMENTS

Our thanks to Canada's Natural Sciences and Engineering Research Council for helping to fund this research and to those SIAST students who volunteered to take part in the experiment.

## 6. REFERENCES

[1] Alexander, J., and Cockburn, A. (2008). An empirical characterization of electronic document navigation. *Proceedings of Graphics Interface '08.* Windsor, On., Canada, 123-130.

[2] Alfassi, M. (2004). Reading to learn: Effects of combined strategy instruction on high school students. *Journal of Educational Research*, 97, 171–184.

[3] Anderson, L.W., Krathwohl, D.R. and Bloom, B.S., (Eds), 2000, Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. New York: Longman.

[4] Anderson, R. C., & Pearson, P. D. (1984). A schema-theoretic view of basic processes in reading comprehension. In P. D. Pearson( Ed.), Handbook of reading research (pp. 829-864). New York: Longman.

[5] Azevedo, R., Moos, D., C., Johnson, A., M., Chauncey, A., D. (2010). Measuring cognitive and metacognitive regulatory processes during hypermedia learning: Issues and challenges. *Educational Psychologist*, 45(4), 210-233.

[6] Beck, J. E., Mostow, J., Cuneo, A., & Bey, J. (2003). Can automated questioning help children's reading comprehension? *Proceedings of the Tenth International Conference on Artificial Intelligence in Education* (AIED2003), Sydney, Australia

[7] Bloom, B.J., Englehart, M.D., Furst, M.D., Hill, E.J. and Krathwohl, D.R., (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook 1: Cognitive Domain.* New York: David McKay.

[8] Bransford, J. D., Stein, B. S., Nye, N. J., Franks, J. F., Auble, P. M., Merynski, K. J., & Perfetto, G. A. (1982). Differences in approaches to learning: An overview. *Journal of Experimental Psychology: General*, 3, 390-398.

[9] Fischer, C. (2003). Revisiting the reader's rudder: A comprehension strategy. *Journal of Adolescent & Adult Literacy*, 47, 248–256.

[10] Graesser, A. C., McNamara, D. S. & VanLehn K. (2005). Scaffolding deep comprehension strategies through Point&Query, AutoTutor, and iSTART. *Educational Psychologist*, 40(4), 225-234.

[11] Hammerly, G., and Elkan, C. (2002). Alternatives to the k-means algorithm that find better clusterings. *Proceedings of the ACM Conference on Information and Knowledge Management*, 600-607.

[12] Lloyd, S. P. (1957). "Least square quantization in PCM". Bell Telephone Laboratories Paper. Published in journal much later: Lloyd., S. P. (1982). "Least squares quantization in PCM". IEEE Transactions on Information Theory 28 (2): 129–137.

[13] Mostow, J., & Chen, W. (2009). *Generating Instruction Automatically for the Reading Strategy of Self-Questioning.* Proceedings of the 14th International Conference on Artificial Intelligence in Education, 465-472.

[14] Paris, S. G., & Paris, A. H. (2001). Classroom applications of research on self-regulated learning. *Educational Psychologist*, 36, 89–101.

[15] Razzaq, L., Patvarczki, J., Almeida, S.F., Vartak, M., Mingyu Feng, Heffernan, N.T., Koedinger, K.R. (2009) The ASSISTment Builder: Supporting the life cycle of tutoring system content creation, *Learning Technologies, IEEE Transactions on* , vol.2, no.2, pp.157-166.

[16] Schraw, G., (2010). Measuring self-regulation in computer-based learning environments. *Educational Psychologist*, 45(4), 258-266.

[17] Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., and Robbins, P. (2008). Bloom's taxonomy for CS assessment. *Proceedings of the Tenth Conference Australasian Computing Education*, 155-161.

[18] van Keer, H. (2004). Fostering reading comprehension in fifth grade by explicit instruction in reading strategies and peer tutoring. *British Journal of Educational Psychology*, 74, 37–70.

[19] Wankat, P., and Oreovicz, F., (1993). *Teaching Engineering.* New York: McGraw-Hill.

[20] Yao-Ting, S., Kuo-En, C., & Jung-Sheng, H. (2008). Improving children's reading comprehension and use of strategies through computer-based strategy training. *Computers in Human Behavior*, 24, 1552-1571.

[21] Zywno, M., S. (2003). Hypermedia instruction and learning outcomes at different levels of Bloom's Taxonomy of Cognitive Domain. *Global Journal of Engineering Education*, 7(1), 59-70.

# Model-Based Collaborative Filtering Analysis of Student Response Data: Machine-Learning Item Response Theory

Yoav Bergner[*], Stefan Dröschler[†], Gerd Kortemeyer[‡], Saif Rayyan, Daniel Seaton, and David E. Pritchard
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139

## ABSTRACT

We apply collaborative filtering (CF) to dichotomously scored student response data (right, wrong, or no interaction), finding optimal parameters for each student and item based on cross-validated prediction accuracy. The approach is naturally suited to comparing different models, both unidimensional and multidimensional in ability, including a widely used subset of Item Response Theory (IRT) models which obtain as specific instances of the CF: the one-parameter logistic (Rasch) model, Birnbaum's 2PL model, and Reckase's multidimensional generalization M2PL. We find that IRT models perform well relative to generalized alternatives, and thus this method offers a fast and stable alternate approach to IRT parameter estimation. Using both real and simulated data we examine cases where one- or two-dimensional IRT models prevail and are not improved by increasing the number of features. Model selection is based on prediction accuracy of the CF, though it is shown to be consistent with factor analysis. In multidimensional cases the item parameterizations can be used in conjunction with cluster analysis to identify groups of items which measure different ability dimensions.

## 1. INTRODUCTION

Online courses offer the prospect of large data sets of student responses to assessment activities that occur over time and under varying conditions (e.g. training, practice, graded homework, and tests). These present a more complex analysis task than test data recorded under constrained circumstances, but they offer the opportunity to learn about learning (e.g. over a semester, or from a specific intervening instructional activity) in the spirit of evidence-centered design [1]. Analyzing such data will require extensions of standard assessment methods such as Item Response Theory (IRT), for example when multiple attempts are allowed [2].

In the context of educational measurement, item response models have numerous advantages over classical test theory, and their use is widespread [3]. Despite a variety of available software packages, IRT parameter estimation is still technical and goodness-of-fit analysis continues to be a subject of research [4; 5]. In this paper we describe an alternate approach to IRT parameter estimation and goodness-of-fit motivated by machine learning. Our approach springs from an operationalist interpretation of the goals of IRT as stated by Lord [6]: "to describe the items by item parameters and the examinees by examinee parameters in such a way that we can predict probabilistically the response of any examinee to any item, even if similar examinees have never taken similar items before."

Collaborative filtering (CF) is commonly used in recommender systems with the goal of recommending unfamiliar items to a user based on ratings of those items by other users and prior rating information by the user in question [7]. The Netflix prize, for example, drew much attention to the problem of movie recommendations [8]. The idea behind any collaborative filter is that when multiple users interact with overlapping subsets of items, information from the interactions can be extracted and used to make probabilistic inferences about potential future interactions. Memory-based CFs attempt to do this by exploiting similarity between users based on a vector of their prior interactions. A naive algorithm might predict that user J will have identical interactions to those of the most similar user K (or to cluster of similar users). This descriptive approach does not attempt to model, causally or otherwise, the nature of the individual interactions. By contrast, model-based CF uses the partial interaction information to model a set of parameters for the users and the items which, taken together, can reconstruct probabilistic predictions about the missing interactions. In this aspect, CF and IRT have the same end.

The structural similarity between IRT and logistic regression has been noted in [9; 10]. Beck and Woolf [11] have applied a linear regression machine learning algorithm to an intelligent arithmetic tutor to predict when a student would answer a problem correctly (and in how much time). Desmarais and Pu [12] have compared Bayesian modeling of knowledge spaces to IRT in pursuit of examinee ability estimation. Whereas Bayesian knowledge tracing requires identification of subject-specific knowledge components, multidimensional IRT is a general framework for measuring ability along multiple dimensions.

This paper explores the application of model-based collaborative filtering (CF) to the analysis of student responses with similar goals to IRT, i.e. finding parameters for students

[*]bergner@mit.edu
[†]also: Ostfalia University of Applied Sciences
[‡]permanent affiliation: Michigan State University

and items that combine to predict student performance on an item by item basis. From machine learning, we borrow the notion of learning the model from the data. Rather than assign an item response model *a priori*, we use the CF to train a class of log-linear models on the data and select the one which performs the best in terms of prediction accuracy. The model is selected for capturing maximal information from a student response matrix, with no prior knowledge about the data assumed. We show that several standard IRT models emerge naturally as special cases.

In the remaining sections, we describe the numerical protocol for parameter estimation as well as an approach to goodness-of-fit based on prediction accuracy and cross-validation techniques standard in machine learning. The approach is naturally suited to comparing different IRT models, both unidimensional and multidimensional. We apply the CF to two sets of student response data. One of the two, containing roughly 120 online homework responses in a General Chemistry course with 2000 students, hints strongly at two dimensions of skill and discrimination for students and items respectively. We demonstrate that the items, thus parametrized by the CF, cluster into the same groupings that are suggested by principal component analysis.

# 2. REGULARIZED LOGISTIC REGRESSION AS A COLLABORATIVE FILTER

## 2.1 Parameter Estimation

We describe the collaborative filtering approach for dichotomously scored responses using regularized logistic regression. Particular IRT models obtain as a special case.

A binary classifier of individual responses is built *ab initio* around a logistic function

$$P = \frac{1}{1+e^{-Z}} \tag{1}$$

which provides a mapping from the real line to the probability interval [0,1]. We are given a response matrix $U_{si}$ whose rows represent the response vector of student $s$ to each item $i$. Each student is to be parametrized by a vector $\theta_k$ and each item by a vector $X_k$. The vectors are by design of commensurate dimension (known as the number of features $n_f$) such that a scalar product can be constructed, the logit, or inverse of the logistic function,

$$Z = \theta \cdot X = \sum_k \theta_k X_k \tag{2}$$

Although student and item indices have been suppressed, $Z$ is a matrix product of $\theta$ ($N_s \times n_f$) and $X$ ($n_f \times N_i$ ). It is useful to modify the description slightly to include a bias component (fixed, equal to 1) on either the student side or the item side, or both, by considering generalizations such as

$$\theta^* = \begin{pmatrix} 1 \\ \theta \end{pmatrix} \qquad X^* = \begin{pmatrix} X \\ 1 \end{pmatrix} \tag{3}$$

in which case

$$Z = \theta^* \cdot X^* = X_0 + \sum_k \theta_k X_k + \theta_0 \tag{4}$$

where we have taken the liberty of relabeling the indices for simplicity of presentation. The bias component in a student or item vector does not add parameter information but importantly allows the logit to be a function of the difference between student and item parameters. (Nothing is gained by having more than one bias component since a sum of student or item parameters defines a single alternate parameter with the same information). The logistic function now generates a probability (or expectation) matrix with the dimensions $N_s \times N_i$ of the response matrix $U_{si}$,

$$P_{si} = \frac{1}{1+e^{-Z_{si}}} \tag{5}$$

The likelihood function for the observed response matrix $U$ given the parameters $\theta$ and $X$ is given by the product

$$L(U|\theta, X) = \prod_s \prod_i P_{si}^{U_{si}} (1 - P_{si})^{(1-U_{si})} \tag{6}$$

and remains to be maximized by suitable assignment of student and item parameters. For computational benefit, one typically uses the logarithm of the likelihood function. If we multiply the log likelihood by $-1$ (turning the maximum into a minimum), we can relabel the result in the convention of machine learning as the "cost function"

$$J(\theta, X) = -\sum_s \sum_i [U_{si} \log P_{si} + (1 - U_{si}) \log(1 - P_{si})] \tag{7}$$

Numerically maximizing the likelihood function $L$ or (equivalently) minimizing the cost function $J$ is quite fast on a modern desktop with off-the-shelf optimization packages (in our R implementation, we use `optim` with method "BFGS"). Typically these min/max finders take as arguments one long parameter vector (formed by unrolling the $X$ and $\theta$ matrices) and a definition of the cost function and its gradient. As of this writing, a response matrix of 2000 students and 50 items takes about 10 seconds to process on a 3.4 GHz Intel i7 machine. This approach to Joint Maximum Likelihood Estimation (JMLE) no longer necessitates a stepwise update of item and student parameters as was once standard [13; 14; 15].

As the number of model features $n_f$ is increased in any data fitting scenario, it becomes possible to minimize the cost function with parameters that do not generalize well to new data, i.e. to over-fit the data. Regularization terms may be introduced in the cost function to reduce over-fitting. To equation 7 we add the terms (sums exclude any bias components)

$$\lambda \sum_{k=1} \theta_k^2 + \lambda \sum_{k=1} X_k^2 \tag{8}$$

where the optimal regularization parameter $\lambda$ can be determined from cross-validation as discussed is section 2.3.

## 2.2 IRT Recovered as Special Cases of the CF

It is now possible to show explicitly how IRT models emerge from this framework. To keep track of the absence or presence of the optional bias component, we label the dimensionality of the student or item vector as an ordered pair. The first component refers to the number of information-carrying

parameters while the second (either a zero or a one) indicates whether or not a bias component is used. Thus the Rasch model (operationally equivalent to the 1PL model) obtains under the arrangement

$$\left.\begin{array}{ll}\dim(\theta)=(1,1) & \theta^*=(1\ \theta)\\ \dim(X)=(1,1) & X^*=(X\ 1)\end{array}\right\}\rightarrow Z_{\mathrm{Rasch}}=X+\theta \quad (9)$$

where we have used the generalized form of the logit constructed in equation 4. The scalars $\theta$ and $X$ here are identified with the student ability and item easiness parameters in the Rasch model.

The Birnbaum 2PL model, still unidimensional in skill, is obtained as

$$\left.\begin{array}{ll}\dim(\theta)=(1,1) & \theta^*=(1\ \theta)\\ \dim(X)=(2,0) & X=(X_1\ X_2)\end{array}\right\}\rightarrow Z_{\mathrm{2PL}}=X_1+\theta X_2 \quad (10)$$

Although the slope-intercept form of the logit appears in the literature, it is common to map $X_1$ and $X_2$ to the discrimination and difficulty parameters $\alpha$ and $\beta$, where $\alpha = X_2$ and $\beta = -X_1/X_2$, such that $Z = \alpha(\theta - \beta)$.

As a final example, Reckase and McKinley [15; 16; 17] have defined as M2PL the multidimensional extension of the 2PL model for $m$ skill dimensions, which emerges here when

$$\left.\begin{array}{ll}\dim(\theta)=(m,1) & \theta^*=(1\ \theta_1\ldots\theta_m)\\ \dim(X)=(m+1,0) & X=(X_0\ X_1\ldots X_m)\end{array}\right\}\rightarrow Z_{\mathrm{M2PL}}=X_0+\sum_{i=1}^{m}\theta_m X_m \quad (11)$$

This is a *compensatory* multidimensional model to the extent that high values of one component of $\theta$ may compensate for low values in another component. However the model is still capable of describing items which have very low discrimination along one or more skills. The $X_m$ item parameters for $m > 1$ should be seen as "discrimination-like" parameters whereas a "difficulty-like" parameter along each axis could be constructed by analogy with the 2PL model as the ratio $-X_0/X_m$.

## 2.3 Evaluating the Model, or Goodness-of-Fit

The CF minimization procedure results in a set of parameters for each student and item. These can be used to construct item response curves (or surfaces or hyper surfaces) as a prelude to studying model-data fit. An alternate approach however, common to machine learning algorithms, is to sequester a portion of the response matrix as a test set which is not considered during parameter estimation. Once parameters are estimated using the remaining "training" data, these same parameters are used to predict the values in the test set (where a probability value of greater than 0.5 results in the prediction of a correct item response). The percentage of correctly classified elements is the accuracy score. An intermediate test-set can be used for cross-validation, for example to adjust the regularization parameter to avoid over-fitting the training set. Moreover by subsampling multiple times (either with disjoint partitions or random subsamples) and averaging the accuracy score, subsampling variability can be controlled.

In the following section, we present results of data analyzed using this recipe, with 30% of the response matrix randomly subsampled for use as a test set, repeating 100 times. In section 4, we discuss interpretation of the accuracy score as a goodness of fit statistic.

# 3. SAMPLE RESULTS OF CF ANALYSIS

We analyze three data sets, two real and one simulated. The first set comes from a pre-test administration of a physics instrument, the Mechanics Baseline Test (MBT) [18] at the Massachusetts Institute of Technology over multiple years from 2005-2009 (26 items and 2300 examinees). The MBT is a standard instrument used to gauge student learning gains on and competencies with essential concepts in introductory physics. A superset of these data has been described and analyzed by Cardamone et al. using (unidimensional) IRT [19].

To test whether the collaborative filter would indeed "discover" multidimensionality of skills in student response data, we constructed a second data set of simulated responses to a two-skill test, assuming correlated skill-components but unidimensional items. In other words, 2000 skill-pairs were sampled from a multivariate Gaussian distribution and a response matrix for 60 items simulated based on a 2PL unidimensional model. Responses to the first 30 items depended only on the first skill component, while responses to the last 30 items depended only on the second component. The two skills over the sampled population were correlated with a Pearson coefficient $r = 0.58$.

The third data set comes from online homework data using LON-CAPA for a General Chemistry class at Michigan State University (MSU). The class was selected for study because it had a large student enrollment in a typical year (N = 2162), and because the 120 items were repeatedly administered over several years between 2003-2009. Although students were allowed multiple attempts on homework problems, the responses were scored correct/incorrect on first try for this analysis. No prescreening of the items was performed, and the data analysis was completely blind to the content of this course.

When the dichotomously scored response matrix contained omitted responses (up to 40% in the General Chemistry homework) the sum over matrix elements in Equation 7 and the computation of the accuracy score both excluded omitted responses.

For each data set, the model space was scanned by starting with $\dim(X) = (1\ 0), \dim(\theta) = (1\ 0)$ and proceeding incrementally subject to the commensurability constraint (i.e. to construct a scalar product of $\theta$ and $X$). In the figures below we denote each model by combining the dimensions of $\theta$ and $X$ into one compact string $(\dim(\theta)\ \dim(X))$, i.e. (1010). In this notation, the model (2130) is read as containing two skill parameters plus a bias parameter and three item parameters (no bias). The apportionment of bias parameters means that both skill parameters multiply an item parameter, but there is one item parameter that remains as a term by itself in the logit.

Figures 1-3 display the accuracy scores of the CF models as the dimensionality is varied. For reference, we indicate with shaded regions the separation of the model space by the dimensionality of student skills. We also indicate with vertical dashed lines the CF models corresponding to particular IRT models. We observe that for the MBT data set, accuracy increases up to the unidimensional 2PL model, but

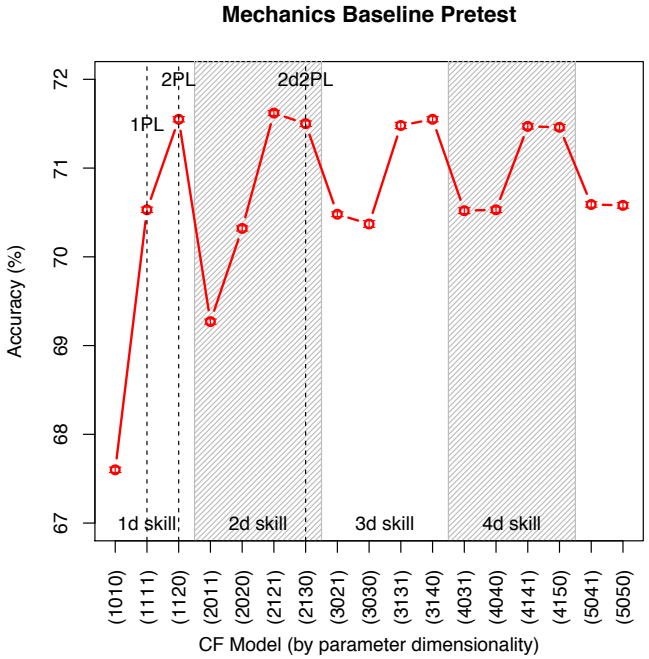



Figure 1: Model by model accuracy scores using the Mechanics Baseline Test data. Performance peaks at the 2PL model and is not improved by additional features.

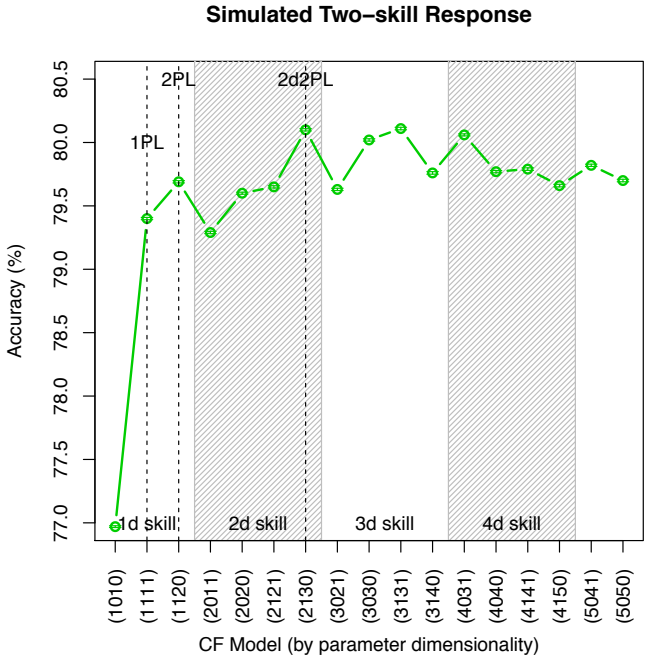



Figure 2: Model by model accuracy scores using the simulated two-skill responses. Two-dimensional models (and the 2d-2PL model in particular) perform optimally.

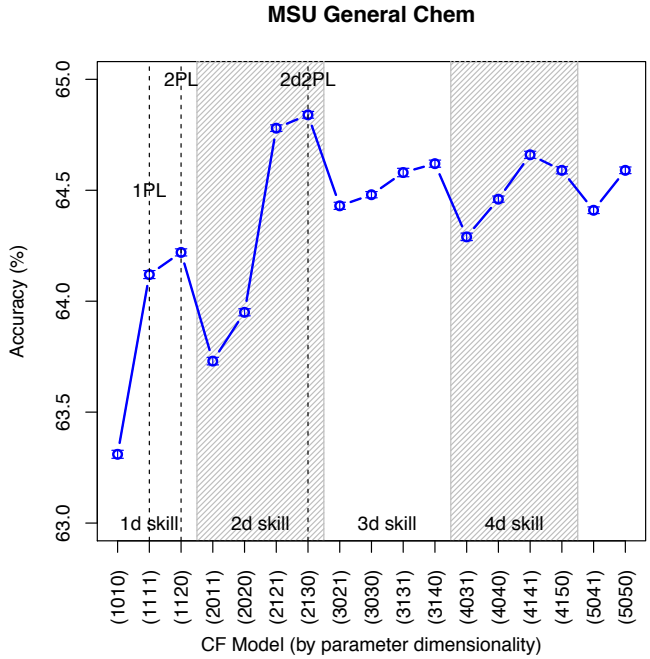
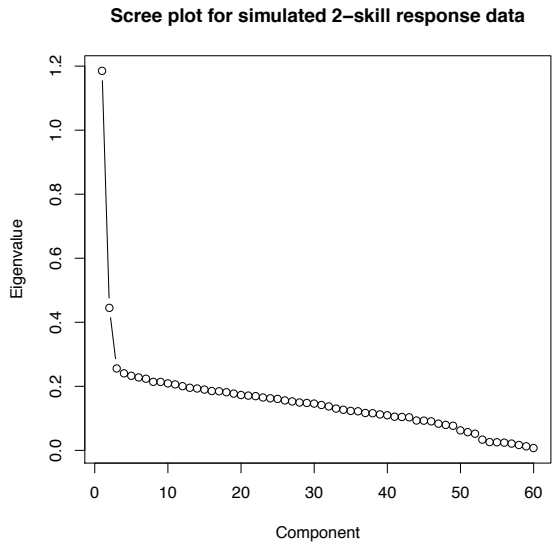



Figure 3: Model by model accuracy scores using online General Chemistry homework. Two-dimensional models (and the 2d-2PL model in particular) outperform unidimensional models.

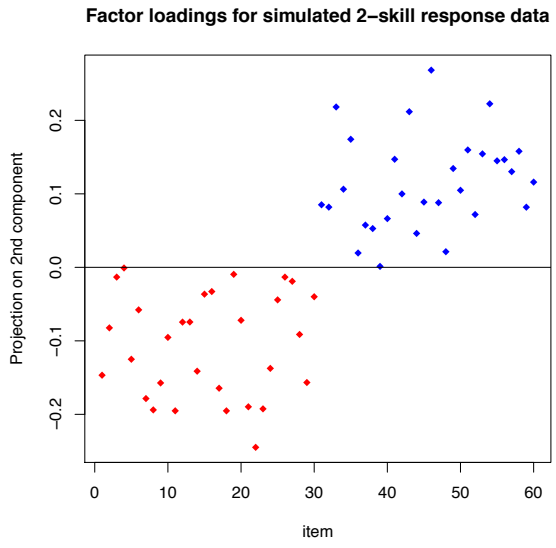
no significant gains are achieved by going to higher dimensional models. In the simulated response data based on a two-part, two-skill test, an accuracy improvement is realized by going to two-dimensional models (and the 2D-generalized 2PL MIRT model in particular), but again this asymptotic limit is not exceeded by higher-dimensional models. This is not surprising given that the simulated data were devised using two skills, but it serves as confirmation that the CF is capable of learning this feature of the data. The substantive result is that the General Chemistry analysis (Fig. 3) follows the pattern of the two-dimensional simulated data and not the unidimensional MBT data.

We note that among the four possible models representing $m$ skill dimensions (for $m > 1$) the latter two models appear to outperform the first two (except in the case of the simulated data). The better performing models are the Reckase M2PL model (m 1 m+1 0) and a hybrid model (m 1 m 1) which could be thought of as M2PL along all but one skill component and 1PL for the remaining skill. Models with higher dimensionality require larger regularization parameters to avoid over-fitting. The apparent degradation of performance for increasing dimensionality is most likely due to over-fitting/sub-optimal choice of regularization parameter (the choice was suitable for the MBT data).

To understand the structure of the simulated two-dimensional data set and calibrate our perceptions for the General Chemistry data, we perform an exploratory factor analysis of the simulated response matrix and show the scree plot in figure 4(a). We plot the projection of each item (factor loading) onto the second principal component in figure 4(b). Whereas

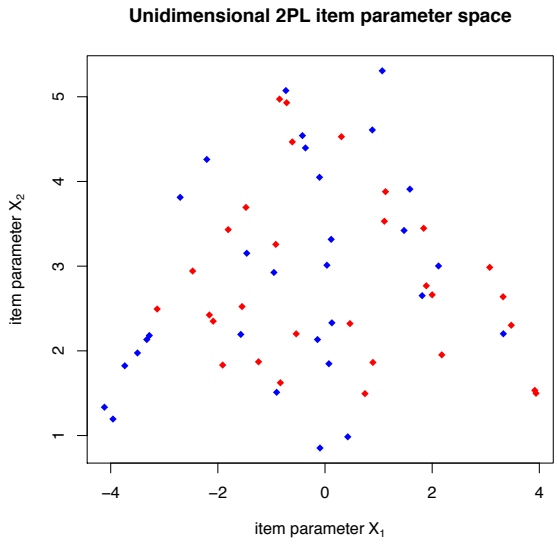



(a)

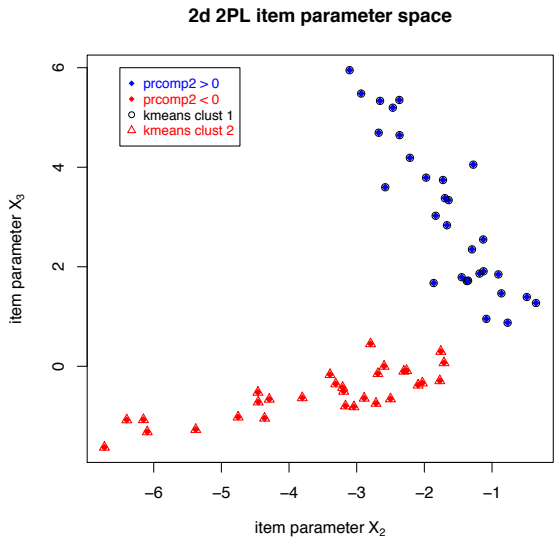



(b)

Figure 4: Simulated two-skill analysis: (a) scree plot and (b) projection of items onto second principal component for simulated data set. Color is added in (b) to identify points in later figures.




(a)




(b)

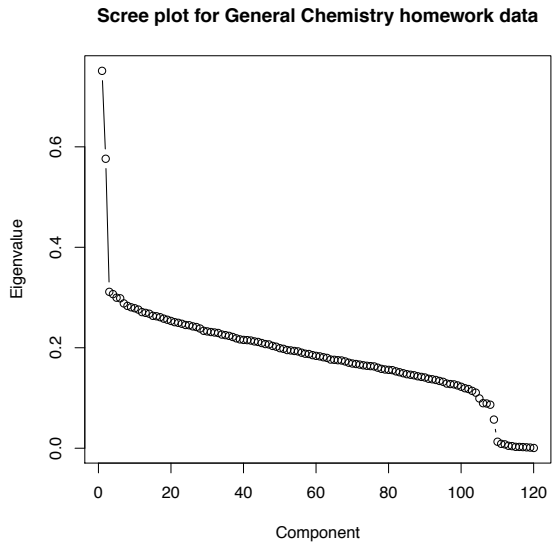
Figure 5: Simulated two-skill analysis: Item parameter space in (a) 1d- and (b) 2d-2PL IRT. Color coding is based on second principal component loading, and $k$-means cluster analysis is superimposed using shapes in (b).

the first principal component captures the variance in overall score or (unidimensional) skill, the second component will differentiate between students who may have the same overall score but perform proportionately better or worse on different groups of items.

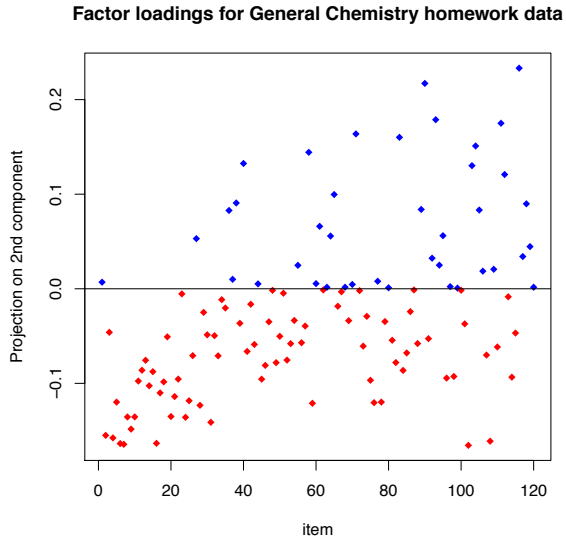
The second principal component loadings in figure 4(b) clearly distinguish two different subsets of items in the simulated data, the first and second half of the item set *by design*.

In figures 5(a) and figure 5(b), we now plot the items as points in the item parameter space generated by two CF models: the (1120) CF model corresponding to unidimensional 2PL (the full item-parameter space is 2-dimensional, spanned by $X_1$ and $X_2$) and the (2130) CF model, corresponding to 2d-2PL IRT. There are three item parameters in the latter model, and we examine the reduced parameter space spanned by the two discrimination-like parameters $X_2$ and $X_3$. The parameters plotted here come from a single run of the CF algorithm.

The unidimensional model blurs any distinction between the two known groups of items, but this distinction is manifest in the 2d-2PL model. The roughly orthogonal arms in figure 5(b) reflect the fact that in our simulated responses, each

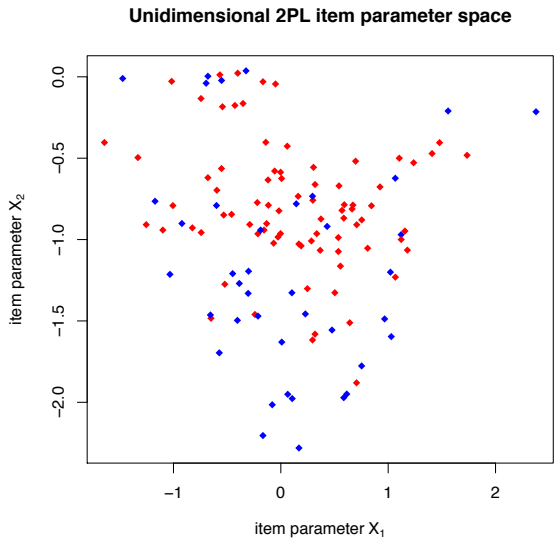



(a)

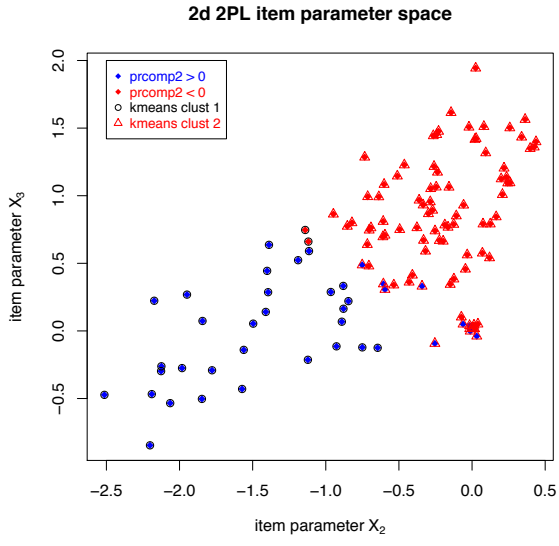



(b)

Figure 6: General Chemistry homework data justifies two-dimensional models. Scree plot (a) showing two significant eigenvalues and (b) projection of items onto second principal component. Color is added in (b) to identify item points in later figures.




(a)




(b)

Figure 7: General Chemistry: (a) unidimensional 2PL item parameter space shows little separation of colors (corresponding to loading onto the second principal component). Conversely (b) 2d-2PL IRT clearly separates color-coded items in the space of two discrimination-like parameters; moreover cluster analysis in this space identifies the border.

item was truly unidimensional in skill-dependence and thus does not discriminate at all with respect to the complementary second skill. We superimpose the results of a $k$-means (with $k = 2$) clustering analysis indicated by shape on the plotted points in figure 5(b). All the red (blue) points are overlapped with triangles (circles), showing that the clustering algorithm finds the same two groups that were identified by the factor loading in figure 4(b). We have verified that this clustering is manifest in three dimensions as well using a 3d-2PL model on simulated data.

We repeat the procedure to visualize the results of the General Chemistry data in figures 6(a)-7(b) with similar results. The cluster analysis in the 2d-2PL parameter space identifies the same two groups as the principal component analysis for over 90% of the items (discrepant items are those that fall very close to the zero line in figure 6(b)).

We emphasize that the choice of model (2130) was driven by the accuracy score of the CF algorithm. Given the model, the two clusters of items emerge from the assignment of

discrimination-like item parameters which best predict the response matrix. We have added no information about the items nor offered any interpretation of the meaning of the two clusters in this case, though we are working with domain experts on identifying the significance. For the simulated data, the two clusters emerge as expected from simulated responses predicated on the assumption that the different item groups test different, though correlated, abilities of the examinees.

# 4. INTERPRETATION OF THE ACCURACY SCORE STATISTIC

It may be noted that the overall accuracy scale differs in each of the figures 1-3, that the scores sometimes seem rather unimpressively low ($\approx$ 65%), and that in some cases the model scores for a given data set differ by only a fraction of one percent. Since we claim that this score provides a basis for preferring one model over another, it behooves us to discuss the meaning of the score value itself.

Beck and Woolf also observed that in any probabilistic binary classifier, the maximum expected accuracy score depends on the distribution of values in the probability (or expectation) matrix [11]. For example, if all probabilities (for each student-item pairing) are equal to 0.75, then all responses would be predicted by the binary classifier to be correct, though of course only 75% should be expected. Perhaps less intuitive, if the values in the probability matrix are distributed uniformly over all values in the interval $[0, 1]$, the expected accuracy score will also be 75%.

A workaround suggested in [11] is to bin the matrix elements into probability bins before comparing with the observed responses. This indeed results in a visible one-to-one correspondence between expected bin-fractions and observed bin-fractions, but bin-based statistics inevitably raise several concerns about the binning procedure itself. Certainly binning choice is not a characteristic of the model. Instead, we probe the accuracy score formally as follows. If the distribution of $p$ values in the expectation matrix is given by a distribution function $g(p)$, then the expected accuracy score is given by the following "average"

$$S = \int_0^{0.5} (1-p)g(p)dp + \int_{0.5}^{1} pg(p)dp \qquad (12)$$

where the first term accounts for predicted-to-be-wrong and the second term for predicted-to-be-right matrix elements. The shape of $g(p)$ in turn depends on the distribution of the student and item parameters and the function that is used to model the probability. As an explicit example, for the Rasch or 1PL model, the probability of a correct response when the student skill is $\theta$ and the problem difficulty is $\beta$ is given by

$$p = \frac{1}{1 + e^{-(\theta-\beta)}} \qquad (13)$$

If student skills are distributed as $g_\theta(\theta)$ and item difficulties as $g_\beta(\beta)$, then $g(p)$ can be shown to be the convolution

$$g(p) = \frac{1}{p(1-p)} \int_{-\infty}^{\infty} g_\theta(\theta) g_\beta \left(\theta + \ln \frac{1-p}{p}\right) d\theta \qquad (14)$$

Although the model dependence has been folded into equation 14, the dependence on the distribution of item difficulties is explicit. The accuracy score thus cannot be meaningfully compared for two different data sets unless the examinees and items are drawn from very similar distributions. For 2PL and M2PL models, the best score will also be a function of the distribution of item discriminations. In fact, we have observed that after removing two MBT items with pathological item response curves found in [19], prediction accuracy on the remaining data increased by 2 percentage points, while this gain was not observed when two randomly selected problems were removed.

In view of the model dependence of equation 14, a cautionary flag might be raised in using the accuracy score to compare different models on a given data set. However since the models are designed to predict the data, we argue that this model-dependence is justly accounted for in using the accuracy score as a goodness-of-fit statistic.

In practice it is much easier to calculate the expected score in equation 12 numerically from the expectation matrix without any integrals. Simply replace all probabilities less than 0.5 by one minus the probability and average over the resulting matrix.

# 5. SUMMARY AND CONCLUSIONS

We have applied a model-based collaborative filter, i.e. a numerical method for analyzing a dichotomous student response matrix with the goal of predicting the observed responses. Relying on readily available optimization code, the CF is fast, flexible and stable. We showed that CF naturally parameterizes a series of models with increasing dimensionality and that this family contains several common unidimensional and multidimensional IRT models.

We showed with sample data that the CF can aid in model-selection and that the multidimensional-model capability can result in improved prediction accuracy and easy investigation of whether the data are better fit by alternate models. Practitioners of IRT will be pleased to learn that, at least in the cases considered here, CF was not able to improve significantly on the quality of fit achieved using standard, but in two cases multidimensional, IRT models. Moreover, the dimensionality of models suggested by the CF and the clustering of items in the ensuing parameterizations are consistent with results from exploratory factor analysis.

Finally, the stability, speed, close connection with IRT, and easy generalizability of CF recommends it very highly for use in analyzing student response data of all sorts.

# 6. ACKNOWLEDGEMENTS

We are grateful to MIT and NSF Grant No. DUE-1044294 for supporting this work. Additionally, we thank Bob Field and Sarah Seaton for their help in diagnosing chemistry questions.

# 7. REFERENCES

[1] Robert J. Mislevy and G. D. Haertel. Implications of Evidence-Centered Design for Educational Testing. *Educational Measurement: Issues and Practice*, 25(4):6–20, 2006.

[2] Young-Jin Lee, David J Palazzo, Rasil Warnakulasooriya, and David E. Pritchard. Measuring student

learning with item response theory. *Physical Review Special Topics - Physics Education Research*, 4(1):1–6, January 2008.

[3] Ronald K. Hambleton and Russell W. Jones. Comparison of Classical Test Theory and Item Response Theory and Their Applications to Test Development. *Educational Measurement: Issues and Practice*, 12(3):38–47, October 1993.

[4] Maria Orlando and D Thissen. Likelihood-Based Item-Fit Indices for Dichotomous Item Response Theory Models. *Applied Psychological Measurement*, 24(1):50–64, 2000.

[5] Maria Orlando and David Thissen. Further investigation of the performance of S-X2: An item fit index for use with dichotomous item response theory models. *Applied Psychological Measurement*, 27(4):289–298, 2003.

[6] F.M. Lord. *Applications of item response theory to practical testing problems*. Erlbaum Associates, 1980.

[7] Xiaoyuan Su and Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009(Section 3):1–19, 2009.

[8] J Bennett and S Lanning. The Netflix Prize. *Proceedings of KDD Cup and Workshop*, 2007(10):8, 2007.

[9] A.D. Wu and B.D. Zumbo. Thinking About Item Response Theory from a Logistic Regression Prespective. In Shlomo Sawilowsky, editor, *Real Data Analysis*, pages 241–269. Information Age Publishing, 2007.

[10] Patrick Mair, Steven P. Reise, and Peter M. Bentler. IRT Goodness-of-Fit Using Approaches from Logistic Regression. *Department of Statistics Papers, UCLA*, 2008.

[11] Joseph E Beck and Beverly Park Woolf. High-level student modeling with machine learning. In *Intelligent tutoring systems*, pages 584–593. Springer, 2000.

[12] Michel C. Desmarais and Xiaoming Pu. A Bayesian student model without hidden nodes and its comparison with item response theory. *International Journal of Artificial Intelligence in*, 2005.

[13] Robert J. Mislevy and Martha L. Stocking. A consumer's guide to LOGIST and BILOG. *Applied psychological measurement*, 13(1):57, 1989.

[14] Frank B Baker and Seock-Ho Kim. *Item Response Theory: Parameter Estimation Techniques*, volume 176 of *Statistics*. Marcel Dekker, 2004.

[15] R. L. McKinley and M. D. Reckase. MAXLOG: A computer program for the estimation of the parameters of a multidimensional logistic model. *Behavior Research Methods and Instrumentation*, 15:389–390, 1983.

[16] M. D. Reckase and R. L. McKinley. The discriminating power of items that measure more than one dimension. *Applied Psychological Measurement*, 15(4):361, 1991.

[17] M. D. Reckase. *Multidimensional Item Response Theory*. Springer, 2009.

[18] David Hestenes and Malcolm Wells. A mechanics baseline test. *The physics teacher*, 30(3):159–166, 1992.

[19] Caroline N. Cardamone, Jonathan E. Abbott, Saif Rayyan, Daniel T. Seaton, Andrew Pawl, David E. Pritchard, N. Sanjay Rebello, Paula V. Engelhardt, and Chandralekha Singh. Item response theory analysis of the mechanics baseline test. In *AIP Conference Proceedings*, volume 1413, pages 135–138, February 2012.

# Predicting drop-out from social behaviour of students

Jaroslav Bayer, Hana Bydžovská,
Jan Géryk, Tomáš Obšívač
Computer Systems Unit
Faculty of Informatics, Masaryk University
Brno, Czech Republic
{bayer, bydzovska, geryk,
obsivac}@fi.muni.cz

Lubomír Popelínský
Knowledge Discovery Group
Faculty of Informatics, Masaryk University
Brno, Czech Republic
popel@fi.muni.cz

## ABSTRACT

This paper focuses on predicting drop-outs and school failures when student data has been enriched with data derived from students social behaviour. These data describe social dependencies gathered from e-mail and discussion board conversations, among other sources. We describe an extraction of new features from both student data and behaviour data represented by a social graph which we construct. Then we introduce a novel method for learning a classifier for student failure prediction that employs cost-sensitive learning to lower the number of incorrectly classified unsuccessful students. We show that the use of social behaviour data results in significant increase of the prediction accuracy.

## 1. INTRODUCTION

One of the current trends in higher education is the substantial increase of the first-year students and, consequently, the volume of educational data. Thousands of students are admitted to study at universities every year. They reach interim results, pass or fail at exams, communicate with each other during their studies and many of them fail to finish their study successfully. University staff would like to encourage such students to finish their studies but it is hard to identify them early also because of the huge number of enrolled students. It is important to explore methods that can extract reliable and comprehensive knowledge from the student data that allow prediction of a drop-out with a sufficiently high accuracy.

In this work we utilized student data that have been stored in the Information System of Masaryk University (IS MU), which stores educational data and comprises of all information about students and their studies, about teachers and courses, and also provides examination management tools, excuses registration system, evaluation of on-line tests, and various forms of communication, e.g. discussion boards. We utilized only a subset of information stored in IS MU that is relevant for prediction of the student success, like capacity-to-study test scores, gained credits, average grades, or gender. Data from IS MU are periodically imported to data warehouse Excalibur [3] that combines three main disciplines of data processing—data management, data mining (DM), and visual analytics.

IS MU also stores the complete history of users' requests to the system. Data about students' social behaviour, such as intensity of interpersonal communication or number of mutually shared files, can be observed and stored either immediately, when the particular system function is used, or later from the complete history of users' requests that is present in the form of the system access log. Relations among students (identified from their social behaviour) are main building blocks of a latent social network. With the help of Social Network Analysis (SNA) [4] we compute several new features of a student from the network, for example neighbours characteristics.

In this paper we introduce a novel method for data generation, pre-processing, and educational data mining (EDM) [1; 14; 10] that utilize both the student records and the data about their social behaviour. We show how to predict student drop-out and school failure using DM [7] methods and SNA. We use SNA for creating new study-related features that can help conventional learning methods to increase the accuracy of predicting student performance or detecting a possible drop-out. We intend to build classifiers for early detection and long term prediction of a potential drop-out. The early detection implies a need for the history of data.

Preliminary results for this task were published in doctoral workshop [2]. The highest measured accuracy was above 80% when only student data were employed. We enriched the student data with the data about social behaviour and achieved an increase of the overall accuracy of about 10%. In both cases, the information gain based machine learning (ML) methods generated the most successful classifiers.

Another approach to the prediction of a student study performance that is based on questionnaires can be found in [12]. In [15], a design of a web based system for solving issues related to student performance in higher education is proposed. It utilizes a quality function deployment in combination with DM methods. A novel ML method predicting drop-out in distance higher education from imbalanced datasets is discussed in [9]. It reveals limitations of the existing methods and proposes another approach based on local-cost sensitive techniques. A novel approach to identify factors influencing the student success is discussed in [11]. It focuses on factors available before the beginning of a students degree program suggesting associative rules for subgroup discovery to predict possible drop-outs. A significant improvement of prediction of freshmen drop-out using cost sensitive learning is described in [5]. The highest accuracy of classification was achieved using decision trees. In comparison with our approach of utilizing social behaviour, a combination of data mining methods with natural lan-

guage processing, especially text mining, was employed in [17] to increase the student retention.
In the following section, we introduce the structure of both the student data and the social behaviour data and the necessary preprocessing steps. We describe how we built the social network and applied the analytical methods in Section 2.2. Section 2.3 describes the DM method used for drop-out prediction. In Section 3 we demonstrate the results and the improvement of the classification by measuring the amount of the additional data explored by SNA. Then we show that high-accuracy classifiers can be created for every student regardless of the actual stage of the study. Discussion of results is in Section 4. Finally, we conclude this paper with an overview of the main results and future work in Section 5.

# 2. DATA AND DROP-OUT PREDICTION

## 2.1 Student data

Our research considers bachelor students of Applied Informatics admitted to Faculty of Informatics, Masaryk University in years 2006, 2007, and 2008. For that period we can obtain data that match the whole length of the standard bachelor study, i.e. three years. The year 2006 as the lower bound is set as the year when social behaviour data have been started to collect. We explored only the students that were in contact with the school community. Such students produce social behaviour data characterizing them in the university setting.
We selected only general attributes of studies to be able to apply our approach to students of any faculty. To predict a drop-out through the whole period of the study we collected data snapshots for each term of student studies. The set of attributes can be divided into three categories according to the type: Student-related attributes, Semester-related attributes and Attributes related to other studies.

Student-related attributes comprise of the following:

(1) **gender**

(2) **year of birth**

(3) **year of admission**

(4) **exemption from entrance exam**

(5) **capacity-to-study test score**—a result of the entrance examination expressed as the percentage of the score measuring learning potential

Semester-related attributes are the following:

(6) **the number of finished semesters**

(7) **recognized courses**—the number of related courses finished in other studies

(8) **recognized credits**—the number of credits gained from recognized courses

(9) **credits to gain**—the number of credits to gain for enrolled but not yet finished courses

(10) **gained credits**—the number of credits gained from finished courses

(11) **uncompleted courses**—the number of courses a student has failed to complete

(12) **second resits done**—the number of the utilized second resits. Each student can exercise the right to the second resit for only as many times as the standard length of the study in years increased by one.

(13) **excused days**—the number of days when a student is exused

(14) **average grades**—the average grade computed from all gained grades

(15) **weighted average grades**—average grades weighted by the number of credits gained for courses

(16) **the ratio of the number of gained credits to the number of credits to gain**

(17) **the difference of gained credits and credits to gain**

Because a student can be enrolled in more studies or also on more faculties, we added also attributes related to other studies of the student. This set of attributes consists of the following:

(18) **the number of parallel studies at the faculty**

(19) **the number of parallel studies at the university**

(20) **the number of all studies at the faculty**

(21) **the number of all studies at the university**

Data that consist of values of all attributes characterizing a study in a point of time have been extracted from Excalibur. The data set contained 775 students, 837 studies and 4,373 examples in total—one example per a term, where the number of terms for a student varied from 1 up to 8.

## 2.2 Social behaviour data

The aforementioned set of 775 students is the core of ego-centered social network. We create it from the students plus their direct schoolmates and relations among them. Relations reflects the patterns of social behaviour data. Then we compute new student features from the network structural characteristics and student direct neighbours attributes.
To obtain knowledge concerning a student from perspective of his or her engagement in the school community, we construct a sociogram, a diagram which maps the structure of interpersonal relations. Such social graph allows to find new features by link-based ranking.
There are number of interpersonal ties already evaluated to enhance IS MU full text search. We compute them either online or through system log processing and store them both in the search engine index as a relevant document non-textual tokens and as a part of the user model. These are then used to better order the search results by matching documents (e.g. e-mails, files, courses) related to the respective users [16].

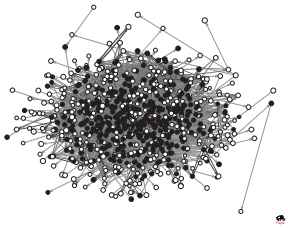


Figure 1: Network with vertices arranged by Kamada-Kawai energy layout algorithm. Dark nodes represent students with successfully finished studies.

Some ties are intuitive and strong facts, namely:

(a) *explicitly expressed friendship*

(b) *mutual e-mail conversation*

(c) *publication co-authoring*

(d) *direct comment on another person*

Weaker ties are more hidden and are derived from the following facts:

(e) *discussion forum message marked as important*

(f) *whole thread in discussion forum or blog marked as favourite*

(g) *files uploaded into someone else's depository*

(h) *assessments of noticeboard's messages*

(i) *visited personal pages*

We measure the value of a tie by its importance and weight it by the number of occurrences. For instance, a tie representing exchange of several e-mails have grater value than a visit to somebody's personal profile. The identification of the best weights is a possible subject of future evaluation. Another notable property of a relation is its direction. It indicates the source and the target of an action which we count as the relation. For example, a person who sent/received an e-mail or who uploaded/received a file in the source/target respectively. Some actions have no direction, e.g. marking the same discussion thread as a favourite one.
As a result we calculated a single number from all mentioned ties reflecting the overall strength of a student relation to any given schoolmate. We found 13,286 such connections representing graph arcs (oriented lines) valued by this number.
Now, the network can be visualized for exploratory analysis of its properties. For example, after applying Kamada-Kawai energy layout algorithm [8] (Fig. 1), we can see that the successful students (black nodes) occupy the area in the middle of the network and are rarely seen on the periphery. In opposite, the unsuccessful ones (white nodes) are placed all over the graph. This, along with results presented later, supports our assumption that higher number and stronger ties have a positive impact on the success of the study while absence of the ties predicts a potential to failure.

#### 2.2.1 New feature generation

This single mode social network of students and their interpersonal ties (i.e. homogeneous information network) allows us to explore it not only visually but also by tools for social network analysis, e.g. Pajek [13]. Moreover, previously unseen features of each student may be computed with such tools. The following two types of features are interesting and give us a new insight into the data.
First, features obtained from the network structure are computed from basic structural characteristics, namely the vertex degrees, the summary of incident line values, and the betweenness centrality:

(22) **degree**—the number of lines that incident with a vertex, represents how many relations the student is involved in

(23) **indegree (or popularity)**—the number of arcs coming to the node, it represents for how many other members of the network the student is a subject of interest

(24) **outdegree**—the number of arcs with opposite direction represents an interest initiated by the given student

(25) **sum of incident line values**—to measure also the strength of the ties

(26) **betweenness centrality**—the number of shortest paths from all vertices to all others that pass through given vertex represents student's importance (global to the network)

Second, features obtained from the neighbourhood properties are also important to examine, and we must measure not only the quantity of person's ties but also their quality. In other words, the academic performance of the surrounding students is important, because it would be hard to get advantage from communication with unsuccessful students. We selected four student features from the data set, preferred by their information gain, to calculate averages of the neighbourhood values (ANV):

(27) **capacity-to-study test score ANV**

(28) **grade average ANV**

(29) **proportion of enrolled and fulfilled credits ANV**

(30) **credits per semester ANV**[1]

### 2.3 Process of drop-out prediction

We aimed at developing an accurate method for drop-out prediction that would also allow predicting the drop-out in an early stage of the study. The method should have minimum of false negatives, i.e. students that have not been recognized to be in danger of dropping-out.
When all the attributes were used the accuracy was poor. That is why we utilized feature selection methods to reduce the dimensionality of the student data extracted from Excalibur data warehouse. We improved the pre-processing method described in [12] by computing the average rank of attributes while eliminating the extreme values.

[1]Surprisingly, when we tried to use these features with weighting using the strength of the corresponding connection, it has not improved the performance of the classifiers.

The goal was to preserve reliability of attributes for classification after the reduction. Therefore we utilized a combination of feature selection/estimation algorithms based on different approaches. We employed three algorithms based on entropy (InfoGainAttributeEval, GainRatioAttributeEval and SymmetricalUncertAttributeEval), an algorithm selecting the minimum-error attribute for prediction (OneRAttributeEval), an algorithm utilizing $\chi^2$-distribution (ChiSquaredAttributeEval), an algorithm preferring attributes highly correlated with the class but with low intercorrelation to others (CfsSubsetEval), an algorithm looking for the smallest subset of attributes having the consistency equal to that of all attributes (ConsistencySubsetEval), and an algorithm assessing attributes by finding the nearest neighbours for a randomly chosen example from every class. It compares the accumulated differences of values of the corresponding features (ReliefFAttributeEval), and we utilized also two filters (FilteredAttributeEval, FilteredSubsetEval). Then, we computed a list of attributes ordered by the average ranks gained from the ordered lists produced by the feature selection algorithms evaluating the significance of the attributes. For every attribute, we skipped the extreme values—the best and the worst evaluations. We reduced the set of attributes to the 22 most relevant and learned the classifiers again. Except for the Naive Bayes (NB) method, all used machine learning methods achieved a higher accuracy. Examples of the removed attributes are the following: being a seminar tutor, the number of password changes, or the number of enrolled courses.

The list of the refined set of attributes in relevance order can be found in Table 1.

Then we computed significant structural characteristics of the social network to gain additional attributes implying social relations among the students.

We employed machine learning methods from Weka on the student data and then on the data that contained also the social behaviour data. To cover all types of machine learning algorithms, we employed J48 decision tree learner, IB1 lazy learner, PART rule learner, SMO support vector machines, and NB classifier. We also employed ensemble learning methods, namely bagging and voting. We utilized cost-sensitive learning (CSM) and then bagging with cost matrix. All methods have been used with default parameter settings. Performance was measured in terms of accuracy (the number of correctly classified examples over the number of all examples) and True Positive Rate (the number of correctly classified examples from the class of unsuccessful students). We used 10-fold cross-validation.

## 3. RESULTS

First we created a classifier using only the social behaviour data but the accuracy did not raise above 69%, in fact, it was lower than for learning from student data. However, if we added the attributes that described the social behaviour to the student data, we observed an increase of accuracy that reached 11%. Main results can be found in Table 2. In the first column represents the results obtained from Excalibur data warehouse, followed by the results for the Excalibur data enriched by the social behaviour data. The baseline was 58.86%. The highest accuracy was obtained with PART, 93.67%, and the True Positive (TP) rate 92.30%. Accuracy for the data without information about student's social behaviour did not overcome 90% and the best result was obtained with decision tree learner, 82.53%, and the TP rate 78.50%.

The most significant attributes include the ratio of the number of gained credits to the number of credits to gain, and the average of this ratio measured for neighbours weighted by the strength of their relation in the social network. The seven most relevant attributes are presented in Table 1.

Table 1: Seven the most relevant attributes

| Order | Avg. Ord. | Attribute |
|---|---|---|
| 1 | 1.000 | (16) |
| 2 | 2.000 | (14) |
| 3 | 2.625 | (15) |
| 4 | 4.500 | (5) |
| 5 | 5.625 | (17) |
| 6 | 6.000 | (8) |
| 7 | 7.750 | (10) |

Table 2: Learning from student data (Excalibur) and student data enriched with social behaviour attributes (With SNA) [%]

| | Excalibur | | With SNA | |
|---|---|---|---|---|
| Method | Accur. | TP | Accur. | TP |
| ZeroR | 58.86 | – | 58.86 | – |
| NB | 77.57 | 73.5 | 72.26 | 83.4 |
| SMO | 79.17 | 64.6 | 81.59 | 74.2 |
| IB1 | 78.14 | 72.5 | 89.80 | 86.2 |
| PART | 82.44 | 73.7 | 93.67 | 92.3 |
| OneR | 75.89 | 57.9 | 88.45 | 83.8 |
| J48 | 82.53 | 78.5 | 89.89 | 88.8 |

We consider social behaviour data to be a characteristic of a student. Therefore, we learned classifiers only from the social behaviour data without snapshots of student studies data. The baseline was slightly lower than for the student data or the enriched data. The most successful classifier was PART with the accuracy 68.82% and the TP with the rate 70.50%. The results are in Table 3.

Table 3: Learning from social behaviour attributes only [%]

| Method | Accur. | TP |
|---|---|---|
| ZeroR | 50.18 | – |
| NB | 64.04 | 80.6 |
| SMO | 63.68 | 83.5 |
| IB1 | 60.10 | 63.5 |
| PART | 68.82 | 70.5 |
| OneR | 59.50 | 57.3 |
| J48 | 68.34 | 65.0 |

Then we analyzed how successful a prediction of a drop-out would be for different time periods. We learned classifiers on interim study results enriched by social behaviour data to recognize drop-outs as soon as possible. Results in terms of accuracy (%) are in Table 4.

Table 4: Learning from student data enriched with social behaviour attributes per semester [%]

| | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7+ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP |
| ZeroR | 50.18 | – | 50.25 | – | 53.87 | – | 58.56 | – | 64.02 | – | 72.20 | – | 76.77 | – |
| NB | 71.45 | 69.1 | 78.87 | 75.8 | 78.98 | 80.7 | 78.77 | 81.8 | 78.66 | 80.2 | 77.56 | 76.3 | 68.60 | 68.0 |
| SMO | 72.40 | 73.9 | 81.33 | 80.2 | 81.02 | 77.5 | 83.22 | 78.1 | 83.74 | 72.3 | 87.56 | 67.5 | 85.48 | 52.3 |
| IB1 | 66.48 | 62.4 | 70.64 | 67.2 | 66.72 | 61.1 | 71.40 | 63.2 | 74.59 | 61.0 | 77.07 | 53.5 | 90.93 | 75.8 |
| OneR | 62.84 | 65.7 | 77.89 | 77.3 | 79.71 | 74.4 | 83.56 | 74.4 | 81.50 | 66.7 | 83.90 | 60.5 | 80.58 | 37.5 |
| PART | 70.13 | 69.5 | 74.82 | 74.3 | 76.20 | 72.8 | 76.20 | 73.1 | 77.24 | 69.5 | 79.51 | 64.0 | 91.11 | 83.6 |
| J48 | 70.73 | 71.2 | 74.82 | 72.8 | 75.77 | 72.5 | 77.91 | 72.7 | 77.64 | 67.8 | 80.00 | 63.2 | 87.11 | 68.8 |

Subsequently, we focused on prediction of drop-outs when the history of data about student studies is employed. All data snapshots were used. Results in terms of accuracy (%) are in Fig. 2. On X axis there is a period of study in semesters (e.g. 3 means that only the data from the first 3 semesters have been used for building the classifier). More details are in Table 5 and Table 6.

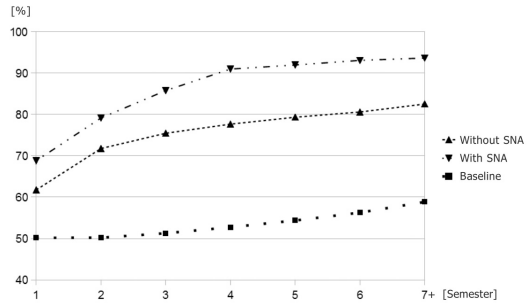



Figure 2: Classifications according to semesters

We can see that for all periods the classification that used only the student data achieves lower accuracy in comparison to the classification on the enriched data. Moreover, starting with the period of the first four semesters the accuracy of classification on the enriched data was higher than 90%. We can conclude that four semesters is a period when our model can predict a drop-out with high probability. We consider this result to be satisfactory. The Masaryk University evaluates the learning potential of students before they are admitted to study.

For our task it is more serious when a student is not recognized to be in danger of a drop-out than the opposite situation. To decrease the number of incorrectly classified unsuccessful students, we tested cost-sensitive learning (CSM) and also bagging, and then bagging with cost matrix, always with the most accurate learning algorithm as the base classifier. In the case of cost-sensitive learning, we set a cost matrix to [0, 1, 0.5, 0] so that the cost of false negative error (i.e. of non-recognized weak students) was twice as high. All the results are in Table 7 in the form of Accuracy (%), TP rate (%), and Incorrectly classified unsuccessful studies (ICUS).

Table 7: Meta-classifiers accuracies

| | Accur. | TP | ICUS |
|---|---|---|---|
| Excalibur (J48) CSM | 80.45 | 85.7 | 258 |
| With SNA (PART) CSM | 92.89 | 92.8 | 129 |
| Excalibur (J48) Bagging | 83.30 | 87.8 | 219 |
| With SNA (PART) Bagging | 96.66 | 96.0 | 55 |

## 4. DISCUSSION

Based on the results, we conclude that a student performance appears to be correlated with the social habits, mainly with the frequency of communication. It supports the hypothesis that students with average results but communicating with students having good grades can successfully graduate with a higher probability than students with similar performance but not communicating with successful students. We identified wrongly classified instances and supplemented them with additional information about specific courses. We found that about one third of students did not complete two particular courses (Automata and Grammars and Specialist English). These findings could be useful in the future work.

Classifiers based on the information gain were the most successful ones. The NB classifier suffered from the strong independence assumption, on our data.

We also combined the two most successful classifiers—J48 and PART—and built a meta-classifier where the prediction was computed as the average of probabilities of particular classifiers. However, the overall accuracy was not higher than that of the best classifier.

We investigated the influence of social behaviour data on the accuracy of classification with respect to the gender of students. The additional data did not increase the accuracy at all. Any classifier did not overcome the baseline 92.11%.

In comparison to [12], we employed social network analysis. They achieved higher accuracy but with more specific attributes obtained from the data that was collected specially for the study. These attributes can not be retrieved from standard school information systems, e.g. smoking habits, the parents' level of education, or the number of siblings.

We investigated the influence of cost sensitive learning on the accuracy of a drop-out prediction. Employing a cost-matrix did not decreased the overall accuracy but slightly improved the TP rate. Using bagging with a cost matrix increased both the accuracy and the TP rate. In the case of classification on the student data, the accuracy remained almost unchanged, but the TP rate increased from 78.5% to 87%. The most significant improvement was achieved in the case of classification on the enriched data. The meta-classifier increased the accuracy to 96.66% and the TP rate to 96%. The number of data snapshots of incorrectly classified unsuccessful students decreased from 146 to 55 in the case of the classification using PART. The number of all data snapshots is 4,373.

Table 5: Learning from student data only according to semester [%]

| | 1 | | 1–2 | | 1–3 | | 1–4 | | 1–5 | | 1–6 | | All | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP |
| ZeroR | 50.18 | – | 50.21 | – | 51.28 | – | 52.74 | – | 54.37 | – | 56.28 | – | 58.86 | – |
| NB | 63.80 | 34.5 | 70.56 | 50.5 | 72.47 | 55.0 | 74.66 | 59.1 | 75.82 | 67.4 | 76.64 | 72.7 | 77.57 | 73.5 |
| SMO | 69.41 | 64.7 | 72.62 | 61.9 | 75.26 | 63.1 | 76.58 | 64.9 | 77.64 | 65.5 | 78.41 | 65.4 | 79.17 | 64.6 |
| IB1 | 62.72 | 61.2 | 66.38 | 66.4 | 69.43 | 67.0 | 70.96 | 68.6 | 72.30 | 68.8 | 74.73 | 70.2 | 78.18 | 72.3 |
| OneR | 55.56 | 41.0 | 64.93 | 68.1 | 70.63 | 76.5 | 74.14 | 79.1 | 75.32 | 76.0 | 75.27 | 70.9 | 75.90 | 57.9 |
| PART | 65.35 | 73.4 | 71.29 | 71.5 | 76.33 | 71.8 | 78.97 | 73.3 | 80.01 | 75.0 | 81.34 | 77.9 | 82.44 | 73.7 |
| J48 | 61.77 | 62.8 | 71.77 | 73.0 | 75.47 | 73.6 | 77.67 | 75.2 | 79.34 | 75.5 | 80.61 | 77.1 | 82.53 | 78.5 |

Table 6: Learning from student data enriched with social behaviour attributes according to semester [%]

| | 1 | | 1–2 | | 1–3 | | 1–4 | | 1–5 | | 1–6 | | All | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP | Accur. | TP |
| ZeroR | 50.18 | – | 50.21 | – | 51.28 | – | 52.74 | – | 54.37 | – | 56.28 | – | 58.86 | – |
| NB | 71.45 | 69.1 | 75.05 | 75.4 | 75.81 | 78.3 | 75.41 | 79.7 | 75.41 | 80.7 | 74.80 | 80.9 | 74.07 | 80.8 |
| SMO | 72.40 | 73.9 | 77.10 | 75.7 | 79.15 | 76.7 | 80.10 | 77.5 | 80.36 | 76.4 | 81.66 | 76.7 | 81.68 | 74.4 |
| IB1 | 66.43 | 62.4 | 67.41 | 63.7 | 70.59 | 67.4 | 76.92 | 73.1 | 81.07 | 76.8 | 83.10 | 79.2 | 90.10 | 86.7 |
| OneR | 62.84 | 65.7 | 69.11 | 67.0 | 74.83 | 74.0 | 81.27 | 79.7 | 83.56 | 81.5 | 82.31 | 79.7 | 88.20 | 83.6 |
| PART | 70.13 | 69.5 | 79.65 | 77.6 | 86.60 | 86.7 | 90.21 | 89.3 | 92.38 | 90.9 | 92.99 | 91.1 | 93.51 | 91.9 |
| J48 | 70.73 | 71.2 | 80.01 | 79.1 | 84.93 | 83.0 | 87.40 | 85.7 | 88.77 | 87.1 | 88.25 | 85.8 | 89.57 | 87.2 |

# 5. CONCLUSIONS AND FUTURE WORK

The main goal of this research was to develop a method for mining educational data in order to learn a classifier to predict the success of a student study and verify the method on real data.

We employed DM and SNA methods to solve the task. We verified the method on students of Faculty of Informatics, Masaryk University but the used data were faculty-independent. Therefore, the method can be used for any unit of a university.

We have shown that structured data gained by means of link-based data analysis increased the accuracy of the classification significantly.

We used only the data that are not specific for a faculty. However, to increase the accuracy of the classification it would be useful to enrich the data with faculty-specific attributes, e.g. information about particular exams that a student passed or failed. Another possible way of future improvement may be to exploit more information from the social network.

Actually, we used only information about a student and his or her direct neighbours. It was intentional because this information is easy to gain and also easy to incorporate into the Information system which is the goal of this research. On the other hand, more complex relations may help further increase the system performance. Data about communication between students and teachers may also be useful. Therefore, we plan to build a heterogeneous [6] network where the vertices will be of more types. Different learning methods can be used then, e.g. multi-label classification.

# 6. ACKNOWLEDGEMENTS

We thank Michal Brandejs and all colleagues of IS MU development team for the support. This work has been partially supported by Faculty of Informatics, Masaryk University.

# 7. REFERENCES

[1] R. Baker and K. Yacef. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining*, 1(1):3–17, 2009.

[2] J. Bayer, H. Bydžovská, J. Géryk, T. Obšívač, and L. Popelínský. Improving the classication of study-related data through social network analysis. In *Proceedings of 7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 3–10. Brno University of Technology, 2011.

[3] J. Bayer, H. Bydžovská, J. Géryk, and L. Popelínský. Excalibur - a tool for data mining. In *Proceedings of the Annual Database Conference - Datakon 2011*, pages 227–228. Brno University of Technology, 2011.

[4] P. Carrington, J. Scott, and S. Wasserman. *Models and methods in social network analysis*. Structural analysis in the social sciences. Cambridge University Press, 2005.

[5] G. W. Dekker, M. Pechenizkiy, and J. M. Vleeshouwers. Predicting students drop out: A case study. In *EDM 2009: Proceedings of the 2nd International Conference On Educational Data Mining. Cordoba, Spain.*, pages 41–50, 2009.

[6] J. Han. Mining heterogeneous information networks by exploring the power of links. In *Proceedings of the 20th international conference on Algorithmic learning theory*, ALT'09, pages 3–3, Berlin, Heidelberg, 2009. Springer-Verlag.

[7] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.

[8] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31:7–15, April 1989.

[9] S. Kotsiantis. Educational data mining: a case study for predicting dropout-prone students. *Int. J. Knowl. Eng. Soft Data Paradigm.*, 1:101–111, January 2009.

[10] A. Kruger, A. Merceron, and B. Wolf. A Data Model to Ease Analysis and Mining of Educational Data. In *EDM2010: Proceedings of the 3rd International Conference on Educational Data Mining. Pittsburgh, USA.*, pages 131–140. www.educationaldatamining.org, 2010.

[11] F. Lemmerich, M. Ifland, and F. Puppe. Identifying influence factors on students success by subgroup discovery. In *EDM2011: Proceedings of the 4th International Conference on Educational Data Mining. Eindhoven, the Netherlands.*, pages 345–346, 2011.

[12] C. Marquez-Vera, C. Romero, and S. Ventura. Predicting school failure using data mining. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *EDM2011: Proceedings of the 4th International Conference on Educational Data Mining. Eindhoven, the Netherlands.*, pages 271–276. www.educationaldatamining.org, 2011.

[13] W. Nooy, A. Mrvar, and V. Batagelj. *Exploratory Social Network Analysis with Pajek*. Structural Analysis in the Social Sciences. Cambridge University Press, 2011.

[14] C. Romero and S. Ventura. Educational data mining: a review of the state of the art. *Trans. Sys. Man Cyber Part C*, 40:601–618, November 2010.

[15] A. Sahay and K. Mehta. Assisting higher education in assessing, predicting, and managing issues related to student success: A web-based software using data mining and quality function deployment. *Academic and Business Research Institute Conference*, 2010.

[16] M. Čuhel, M. Brandejs, J. Kasprzak, and T. Obšívač. Access rights in enterprise full-text search. In *ICEIS 2010: Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 1: Databases and Information Systems Integration*, pages 32–39. INSTICC, Funchal, Portugal, 2010.

[17] Y. Zhang, S. Oussena, T. Clark, and H. Kim. Use data mining to improve student retention in higher education - a case study. In *ICEIS 2010: Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 1: Databases and Information Systems Integration*, pages 190–197. INSTICC, Funchal, Portugal, 2010.

# Searching for Variables and Models to Investigate Mediators of Learning from Multiple Representations

Martina A. Rau
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA
marau@cs.cmu.edu

Richard Scheines
Department of Philosophy
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA
scheines@cmu.edu

## ABSTRACT

Although learning from multiple representations has been shown to be effective in a variety of domains, little is known about the mechanisms by which it occurs. We analyzed log data on error-rate, hint-use, and time-spent obtained from two experiments with a Cognitive Tutor for fractions. The goal of the experiments was to compare learning from multiple graphical representations of fractions to learning from a single graphical representation. Finding that a simple statistical model did not fit data from either experiment, we searched over all possible mediation models consistent with background knowledge, finding several that fit the data well. We also searched over alternative measures of student error-rate, hint-use, and time-spent to see if our data were better modeled with simple monotonic or u-shaped non-monotonic relationships. We found no evidence for non-monotonicity. No matter what measures we used, time-spent was irrelevant, and hint-use was only occasionally relevant. Although the total effect of multiple representations on learning was positive, they also had a negative effect on learning, mediated by a higher error-rate. Our evidence suggests that multiple representations increase error-rate, which in turn inhibits learning. The mechanisms by which multiple representations improve learning are as yet unmodeled.

## Keywords

Model search, variable search, mediation, log data, multiple representations

## 1. INTRODUCTION

Learning processes are complex: many factors influence learning outcomes and the mechanisms by which experimental interventions influence learning are often mysterious. Intelligent Tutoring Systems (ITSs) can easily capture large amounts of data during learning, and combined with sophisticated data mining tools, they have the potential to help understand the mechanisms underlying the effects of successful interventions. Most ITSs are instrumented to collect data on several problem-solving behaviors that might mediate learning, such as error-rate, hint-use, and time-spent [13]. Variables that assess students' problem-solving behaviors have been used to model students' learning [3,8] and to improve ITSs [17]. To make use of the potential of ITS data to gain insights into *why* we see certain learning outcomes, however, we have to overcome difficulties in modeling the mechanisms of learning outcomes. First, we may not adequately understand which variables to use to model these complex relationships. We often assume a linear relationship between measures of learning behaviors and learning outcomes, even though linear relationships may not adequately describe such complex relations [1]. Second, there are a very large number of possible models that describe *how* learning behaviors and learning outcomes relate – how can we know which is the right one? The goal of the present paper is to address both of these important issues using variable search, path analytic modeling, and model search.

Many ITSs use multiple representations to support mathematics learning. Although a vast body of research shows that multiple representations can benefit student learning [2], we know little about the mechanisms that underlie the advantage of learning with multiple representations compared to learning with only a single representation. We investigated the benefits of multiple graphical representations compared to the benefits of a single graphical representation in the context of an ITS – thus enabling us to make use of the rich log data provided in order to investigate the mediating role of student learning behaviors. Specifically, students worked with a Cognitive Tutor for fractions. Cognitive Tutors provide problem-solving tasks and individualized support for students during the learning process [10], and have been shown to lead to significant learning gains in a variety of studies [10,11]. The Fractions Tutor provides error messages tailored to specific misconceptions a student may have. Students can also request a sequence of hints for each step. We chose fractions as the domain for our experiments since fractions instruction typically uses multiple graphical representations such as circles, rectangles, and number lines [12]. Each of these representations emphasizes a different conceptual view on fractions [6] and students need to understand each of these conceptual views [12]. Furthermore, fractions pose a major obstacle for students in the elementary and middle grades [12], such that

understanding mechanisms underlying successful learning is an important educational goal.

We conducted two *in vivo* experiments to investigate the benefits from learning with a version of the Fractions Tutor that uses multiple graphical representations compared to learning with a version of the Fractions Tutor that uses only a single graphical representation. In experiment 1, students worked only with a number line (in the single representation condition), or (in the multiple representations condition) with a variety of graphical representations, including circles, rectangles, and number lines. The representations were relatively static: students could interact with the representations only by entering a number into a text field. The picture updated when the student entered the correct number. In each tutor problem, students solved a fractions problem. For instance, students were asked to add two given fractions and by typing the number of shaded sections into a text field, specifying the numerator of the sum fraction. We crossed these two conditions with a second experimental factor: whether or not students received self-explanation prompts to relate the graphical representations to the symbolic notation of fractions (e.g., ½). For example, students were asked to select "adding the number of shaded sections" to the question of what action with a circle diagram corresponds to adding the numerators using fractions symbols. Results based on an analysis of pretests, immediate posttests, and delayed posttests showed that learners significantly benefited from multiple representations, provided that they were also prompted to self-explain [15].

In experiment 2, we included self-explanation prompts in the single representation condition and in the multiple representations condition. Students in the single representation condition worked either only with a number line, only with a circle, or only with a rectangle. Students in the multiple representations condition received all three graphical representations. In this experiment, the graphical representations were interactive: students could interact with the representations by dragging-and-dropping sections from one representation into another, by using buttons to change number of sections, and by clicking on sections to highlight them. Results based on students' test data confirm the findings from experiment 1: students in the multiple representations condition significantly outperformed students in the single representation condition[1].

We hypothesize that multiple graphical representations result in more successful learning behaviors in the learning phase. We investigated these relationships with the log data that the Fractions Tutor recorded during the learning phases of both experiments. We assume that students who make very few errors, ask for very few hints, and spend very little time per step already have a very good understanding of fractions and will not benefit from working with the Fractions Tutor. On the other hand, inefficient learning such as trial-and-error [4], may manifest themselves in making many errors, asking for many hints, and spending a lot of time per step. We expect that students who show these kinds of unsuccessful learning strategies are not engaging in deep processing of the learning contents and will consequently be less likely to benefit from working with the Fractions Tutor. We hypothesize that the most successful learning behaviors will manifest themselves in moderate levels error-rate, hint-use, and time-spent. This suggests that the relationships between error-rate, hint-use, and time-spent with learning is not simple and monotonic, but rather u-shaped (or inverted u-shaped). We investigated this hypothesis by searching for non-monotonic transformations of our "raw" variables that better predict students' learning than do the raw variables. We then used the best variables in path analysis to investigate the mediating role of error-rate, hint-use, and time-spent on students' benefit from multiple graphical representations.

## 2. DATA SETS

The analyses presented in this paper are based on the data obtained from the two experimental studies just described. Students in both experiments received a pretest on the day before they started to work with the Fractions Tutor. The day after students finished working with the Fractions Tutor, students received an immediate posttest. About one week after the immediate posttest, students were given an equivalent delayed posttest. In experiment 1, the pretest was a shorter version of the posttests, the posttests included more advanced items which required students to transfer the knowledge covered by the tutoring system to novel situations. In experiment 2, all three tests were equivalent (i.e., they contained the same type of items, but with different numbers).

In experiment 1, 110 6$^{th}$-grade students worked with either of four versions of the Fractions Tutor (i.e., with a version that included a single graphical representation without prompts, a single graphical representation with prompts, multiple graphical representations without prompts, or multiple graphical representations with prompts). Students worked with the Fractions Tutor for 2.5 hours of their regular mathematics instruction. The average number of errors made per step, the average number of hints requested per step, and the average time spent per step were extracted from the log data obtained from the tutor sessions. Table I shows the means and standard deviations per condition per and per test. Students had a broad range of prior knowledge: the minimum pretest score was 0.00, and the maximum was 1.00. As shown in Table I, students in the MGR condition with prompts outperformed the other conditions both at the immediate and at the delayed posttest. Since in experiment 1, the pretest was not equivalent to the posttests, the pretest scores are not directly comparable to the posttest scores shown in Table I.

[1] This effect was significant for number line items and conceptual transfer on the delayed posttest.

Table II gives an overview of the tutor log data for each condition. While conditions did not differ with regards to error-rate, students who received self-explanation prompts requested fewer hints than students without prompts. Students in the MGR condition with prompts spent relatively more time per step than students in the other conditions, but the differences were small.

| | SGR w/o prompts | SGR with prompts | MGR w/o prompts | MGR with prompts |
|---|---|---|---|---|
| Pretest | 0.79 (0.14) | 0.70 (0.24) | 0.64 (0.25) | 0.75 (0.21) |
| Immediate posttest | 0.77 (0.16) | 0.70 (0.18) | 0.61 (0.23) | 0.83 (0.15) |
| Delayed posttest | 0.77 (0.19) | 0.74 (0.22) | 0.63 (0.21) | 0.85 (0.12) |

Table I. Means and standard deviations (in brackets) of standardized performance on pretest and posttests from experiment 1 per condition: single graphical representations (SGR) with or without prompts, and multiple graphical representations (MGR) with or without prompts.

| | SGR w/o prompts | SGR with prompts | MGR w/o prompts | MGR with prompts |
|---|---|---|---|---|
| Error-rate | 0.27 (0.15) | 0.37 (0.17) | 0.31 (0.12) | 0.34 (0.13) |
| Hint-use | 0.13 (0.31) | 0.04 (0.05) | 0.19 (0.32) | 0.04 (0.09) |
| Time-spent | 10.37 (4.98) | 8.47 (6.77) | 11.93 (10.18) | 13.99 (18.46) |

Table II. Means and standard deviations (in brackets) of error-rate (# per step), hint-use (# per step), and time-spent (in sec) per condition: single graphical representations (SGR) with or without prompts, and multiple graphical representations (MGR) with or without prompts.

In experiment 2, 290 4th- and 5th-grade students worked on one of two versions of the Fractions Tutor (i.e., SGR with prompts, or MGRs with prompts) for about 5 hours of their regular mathematics instruction. As in experiment 1, we extracted the average number of errors made per step, the average number of hints requested per step, and the average time spent per step from the log data. Table III summarizes students' performance on each test for each condition in experiment 2. Again, students started with a broad range of prior knowledge: the minimum pretest score was 0.06, and the maximum pretest score was 0.96. Students in the MGR condition perform slightly better than students in the SGR condition at the immediate and at the delayed posttest. Since in experiment 2, the pretest was equivalent to the posttests, we can compare the pretest scores to the posttest scores: students' average scores improved from pretest to the posttests (see Table III). Table IV shows that students in the MGR condition make slightly more errors and ask for slightly more hints, while spending the same time per step as students in the SGR condition. As in experiment 1, the differences between conditions on the log data variables are small.

| | SGR | MGR |
|---|---|---|
| Pretest | 0.54 (0.23) | 0.57 (0.21) |
| Immediate posttest | 0.60 (0.23) | 0.63 (0.21) |
| Delayed posttest | 0.62 (0.23) | 0.67 (0.20) |

Table III. Means and standard deviations (in brackets) of standardized performance from experiment 2 per condition and test: single graphical representations (SGR) and multiple graphical representations (MGR).

| | SGR | MGR |
|---|---|---|
| Error-rate | 0.14 (0.07) | 0.16 (0.08) |
| Hint-use | 0.04 (0.06) | 0.06 (0.09) |
| Time-spent | 0.14 (0.04) | 0.14 (0.05) |

Table IV. Means and standard deviations (in brackets) of error-rate (# per step), hint-use (# per step), and time-spent (in sec) per condition: single graphical representations (SGR) and multiple graphical representations (MGR).

## 3. DEFINING VARIABLES WITH WHICH TO INVESTIGATE MEDIATORS

In order to investigate whether a u-shaped, non-monotonic relationship between error-rate, hint-use, and time-spent with students' learning describes the association between problem-solving behavior and learning better than the monotonic relationship, we first conducted a search for a non-monotonic transformation that best predicts students' learning using the data from experiment 2. We used a simple algorithm which computed the "optimal level" of error-rate, hint-use, and time-spent by searching for the highest correlation with learning gains from pretest to the immediate posttest, and from pretest to the delayed posttest, respectively. The algorithm used intervals that varied in size and position. For each interval, we computed a binary variable that for each student indicated whether his/her error-rate (or hint-use, or time-spent) was within the interval or outside the interval. We then computed the correlation of this variable with students' learning gains. For the interval that had the highest correlation with students' learning gains, we identified the mid-point as the "optimum" level of error-rate, hint-use, and time spent. Next, we created two new, non-monotonic predictor variables for error-rate, hint-use, and time-spent, respectively: distance from the optimum, and squared distance from the optimum.

To evaluate whether the non-monotonic variables more accurately predict students' learning, we conducted step-wise regression analyses separately for error-rate, hint-use, and time-spent on both the immediate and the delayed posttests. We entered pretest performance, error-rate, hint-use, or time-spent, and the interaction of pretest performance with error-rate, hint-use, or time-spent as predictors into the regression model. Table V provides a summary of the results from the stepwise regression analyses for error-rate. The regression models with error-rate show that the regression models using monotonic variable explain more variance than the non-monotonic variables. Similarly, the best models with hint-use using the monotonic variable explain more variance than the best models with the non-monotonic variables. The most successful regression models with time-spent take only pretest performance into account; neither the monotonic variable for time-spent nor the non-monotonic variables for time-spent were significant predictors.

| | | pre | pre + errors | pre + errors + errors*pre |
|---|---|---|---|---|
| mono-tonic | IP | $\beta_1$ = 81*, $R^2$ = .66 | $\beta_1$ = .81*, $\beta_2$ = -.27*, $R^2$ = .70 | ***$\beta_1$ = .46*, $\beta_2$ = -.48*, $\beta_3$ = .19*, $R^2$ = .71*** |
| | DP | $\beta_1$ = .80*, $R^2$ = .65 | ***$\beta_1$ = .65*, $\beta_2$ = -.24*, $R^2$ = .68*** | $\beta_1$ = .54*, $\beta_2$ = -.38*, $\beta_3$ = .13(*), $R^2$ = .68 |
| distance from optimum | IP | $\beta_1$ = .81*, $R^2$ = .66 | $\beta_1$ = .76*, $\beta_2$ = -.17*, $R^2$ = .68 | ***$\beta_1$ = .63*, $\beta_2$ = -.37*, $\beta_3$ = .23*, $R^2$ = .69*** |
| | DP | $\beta_1$ = .80*, $R^2$ = .65 | ***$\beta_1$ = .78*, $\beta_2$ = -.13*, $R^2$ = .66*** | $\beta_1$ = .71*, $\beta_2$ = -.22*, $\beta_3$ = .11, $R^2$ = .66 |
| squared-distance | IP | $\beta_1$ = .81*, $R^2$ = .66 | ***$\beta_1$ = .76*, $\beta_2$ = -.16*, $R^2$ = .68*** | $\beta_1$ = .73*, $\beta_2$ = -.25*, $\beta_3$ = .09, $R^2$ = .68 |
| | DP | $\beta_1$ = .80*, $R^2$ = .65 | ***$\beta_1$ = .77*, $\beta_2$ = -.13*, $R^2$ = .66*** | $\beta_1$ = .77*, $\beta_2$ = -.12, $\beta_3$ = -.01, $R^2$ = .66 |

Table V. Regression with error-rate: standardized regression weights and variance explained by each regression model for performance on immediate posttest (IP) and delayed posttest (DP). The best model is displayed in bold-italics. $\beta_1$ = pretest (pre), $\beta_2$ =error-rate (errors), and $\beta_3$ = errors*pre.

In sum, the results from the stepwise regressions show, the non-monotonic variables do not predict performance on the immediate or the delayed posttest better than the monotonic variables do. For that reason, we decided to use the original, monotonic variables of error-rate, hint-use, and time-spent for the subsequent path analytical analyses.

## 4. HYPOTHESES AND PATH ANALYSIS MODELING

In order to investigate the mechanisms by which the intervention (multiple graphical representations) might have affected learning, we first specified, estimated and tested two path analytical structural equation models [5,20] for each of the two experiments. Structural equation models provide a unified framework within which to test mediation hypotheses, to estimate total effects, and also to separate direct from indirect effects. The models that represented our hypotheses in both experiments were decisively rejected by the data, and in such a case it is not appropriate to use the model to test mediation hypotheses or estimate effects. Our strategy was to use the Tetrad IV program[2] to search for alternative models that are both theoretically plausible and consistent with the data. In this section, we describe the path analytic models that represent our hypotheses, describe the search algorithms we use to search for alternative models, and briefly summarize the results of our search.

### 4.1 Modeling Our Hypotheses

We hypothesized that multiple representations lead to learning via the three different mechanisms discussed above: error-rate, hint-use or time-spent per step. As each of these variables might also be affected by a student's prior knowledge of fractions, our hypothesis included paths from our intervention variables to each of these mediator variables as well as paths from *pretest* to each of these variables. One of the path models we specified to represent and test our hypothesis about mediation for experiment 1 is shown in Fig. 1.[3] Fig. 2 shows one of the models we specified for experiment 2. Each node in the path models refers to a variable in the data set: *mult_rep* = single vs. multiple representations, *se* = self-explanation prompts, *mr*se* is variable representing a intervention interaction, *pre* = pretest, *time*, *errors*, *hints* = average time spent, # of errors, and # of hints requested per step, *post* = performance on the immediate posttest, *delpost* = performance on the delayed posttest. For both experiments, we hypothesize that pretest performance predicts performance on the immediate and on the delayed posttests, as well as error-rate, hint-use, and time-spent.

[2] Tetrad, freely available at www.phil.cmu.edu/projects/tetrad, contains a causal model simulator, estimator, and over 20 model search algorithms, many of which are described and proved asymptotically reliable in [20].

[3] In path models of this type, also called "causal graphs" [20], each arrow, or directed edge, represents a direct causal relationship relative to the other variables in the model. For example, in Fig. 1 the conditions are direct causes of the mediator variables, but only affect the post-test indirectly through these mediators.

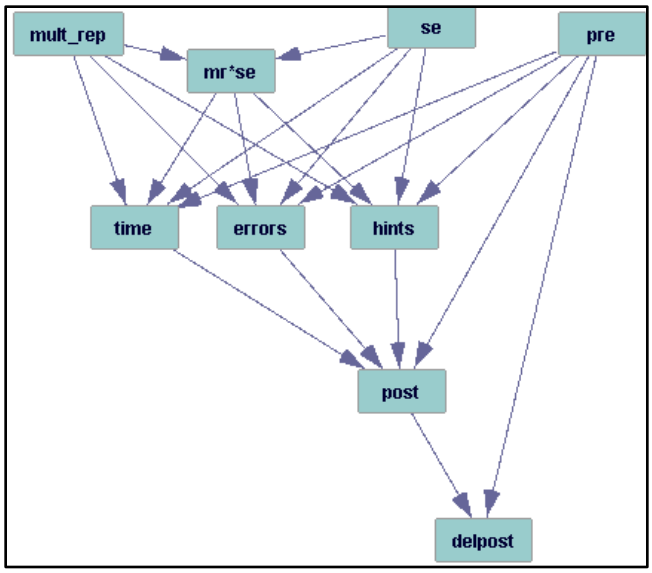



**Fig. 1**. Path model for experiment 1.

In addition, we predict that in experiment 1, multiple representations (*mult_rep*), self-explanation prompts (*se*), and the interaction between multiple representations and self-explanation prompts (*mr*se*) predict error-rate, hint-use, and time-spent. In other words, we predict that the effects of the intervention variables are entirely mediated through students' learning behaviors. Similarly, for experiment 2, we predict that the effect of multiple representations (*mult_rep*) predicts error-rate, hint-use, and time-spent, which corresponds to a full mediation of the intervention through learning behaviors. Hence, the path model for experiment 2 corresponds to the one shown in Fig. 1, except that self-explanation prompts (*se*), and the interaction between multiple representations and self-explanation prompts (*mr*se*) were not present in experiment 2.

Using normal theory maximum likelihood to estimate the parameters of these models, we find that in each case the deviation between the estimated and the observed covariance matrix is too large to be explained by chance (for the model for experiment 1 in Fig. 1: $\chi^2 = 53.8$, df = 16, $p < .0001$,[4] and for the model for experiment in Fig. 2: $\chi^2 = 59.41$, df = 6, $p < .0001$), thus the models do not fit the data and the parameter estimates cannot be trusted.[5]

## 4.2 Model Search

To search for alternatives, we used the GES algorithm in Tetrad IV along with background knowledge constraining the space of models searched [7] to those that are theoretically tenable and compatible with our experimental design. In particular, we assumed that our intervention variables are exogenous, that in experiment 1 our intervention variables are causally independent but direct causes of the interaction variable, that the pretest is exogenous and causally independent of intervention, that the mediators are prior to the immediate posttest and to the delayed posttest, and that the immediate posttest is prior to the delayed posttest. Even under these constraints, there are at least $2^{32}$ (over 4 billion) distinct path models of experiment 1 that are consistent with our background knowledge, and $2^{25}$ (over 33 million) distinct path models of experiment 2.

The qualitative causal structure of each of these linear structural equation models can be represented by a Directed Acyclic Graph (DAG). If two DAGs entail the same set of constraints on the observed covariance matrix,[6] then we say that they are empirically indistinguishable. If the constraints considered are independence and conditional independence, which exhaust the constraints entailed by DAGs among multivariate normal varieties, then the equivalence class is called a *pattern* [14,20]. Instead of searching in DAG space, the GES algorithm achieves significant efficiency by searching in pattern space. The algorithm is asymptotically reliable,[7] and outputs the *pattern* with the best Bayesian Information Criterion (BIC) score.[8] The pattern identifies features of the causal structure that are distinguishable from the data and background knowledge, as well as those that are not. The algorithm's limits are primarily in its background assumptions involving the non-existence of unmeasured common causes and the parametric assumption that the causal dependencies can be modeled with linear functions.

[4] The usual logic of hypothesis testing is inverted in path analysis. The p-value reflects the probability of seeing as much or more deviation between the covariance matrix implied by the estimated model and the observed covariance matrix, conditional on the null hypothesis that the model that we estimated was the true model. Thus, a low p-value means the *model* can be rejected, and a high p-value means it cannot. The conventional threshold is .05, but like other alpha values, this is somewhat arbitrary. The p-value should be higher at low sample sizes and lowered as the sample size increases, but the rate is a function of several factors, and generally unknown.

[5] We also tested variations of these models in which we added direct paths from the condition variables to the post-test and delayed post-test. These variants are also clearly rejected by our data.

[6] An example of a testable constraint is a vanishing partial correlation, e.g., $\rho_{XY.Z} = 0$.

[7] Provided the generating model satisfies the parametric assumptions of the algorithm, the probability that the output equivalence class contains the generating model converges to 1 in the limit as the data grows without bound. In simulation studies, the algorithm is quite accurate on small to moderate samples.

[8] All the DAGs represented by a pattern will have the same BIC score, so a pattern's BIC score is computed by taking an arbitrary DAG in its class and computing its BIC score.

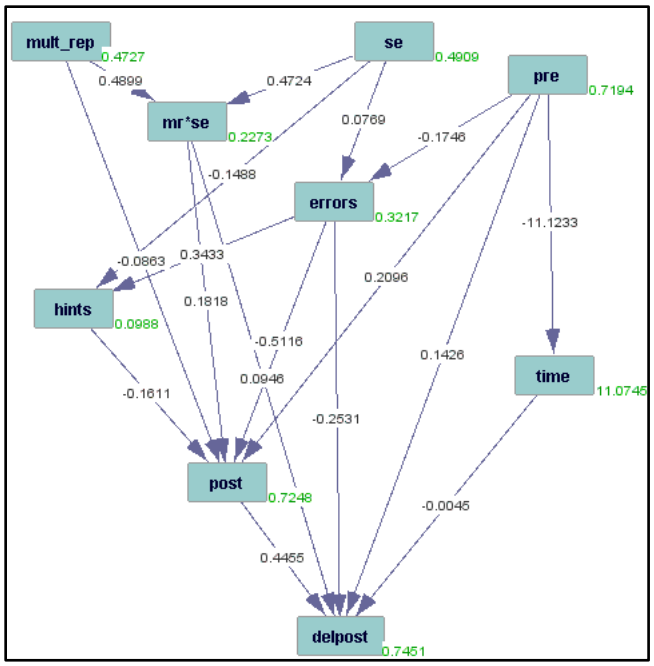



**Fig. 2**. The model found by GES on data from experiment 1, with parameter estimates included. This model fits the data well: $\chi^2 = 22.11$, df = 19, $p = .29$.

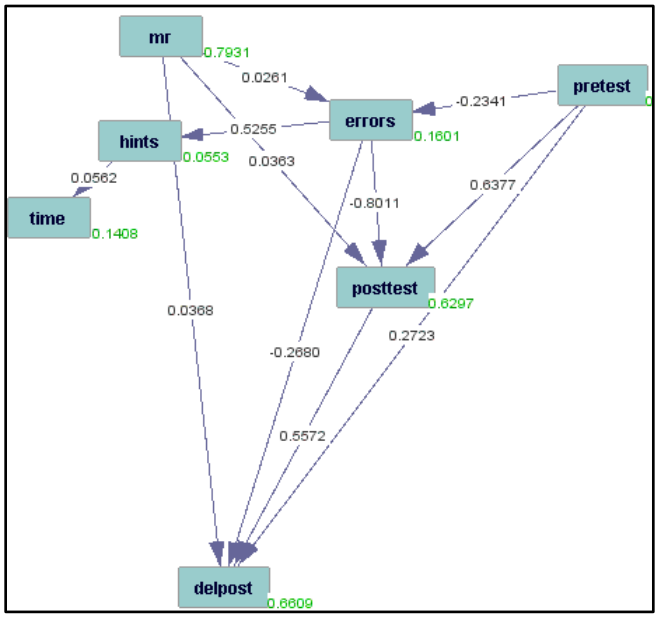



**Fig. 3**. The model found by GES on data from experiment 2, with parameter estimates included. This model also fits the data well: $\chi^2 = 6.89$, df = 10, $p = .74$.

Fig. 2 shows a model found by GES on the data from experiment 1, with path coefficient estimates included. The model fits the data well ($\chi^2 = 22.1$, df = 19, $p = .28$), and contains a number of interesting properties. For one thing, students with higher pretest scores spend much less time per problem, but none of our intervention variables had any influence on time, and the apparent effect of time spent per step during the learning phase is minimal. Multiple representations had a positive effect on learning, but only when self-explanation prompts were also part of the learning environment.[9] Further, there is no evidence that the positive effect of multiple representations is mediated by either error-rate, hint-use, or time-spent. When *not* combined with multiple representations, self-explanation prompts appear to slightly increase error-rate and thus inhibit learning, but slightly decrease hint-use, which, because they appear to inhibit learning, have an overall positive effect on learning.

Fig. 3 shows a model found by GES for experiment 2 that fits the data very well ($\chi^2 = 6.89$, df = 10, $p = .74$). This model indicates that although multiple representations (*mr*) have a positive direct effect on both the immediate posttest and the delayed posttest, they also have a negative indirect effect on both outcomes through error-rate. Learning with multiple representations seems to cause students to make slightly more errors during learning, possibly because the greater variability in tutor problems leads to higher cognitive processing demands. The higher error-rate during the learning phase seems to have a negative influence on test performance. Note that there are two paths from multiple representations to the posttests in the model in Fig. 3, and that the positive direct effect (a bit over 3 ½ percentage points on both) is larger than the indirect negative effect through errors in both cases (2 percentage points on the immediate posttest and about ½ a percentage point on the delayed posttest).

As in experiment 1, hint-use and time-spent do not discernibly mediate the influence of multiple representations on learning. However, students appear to ask for more hints in response to making more errors, and they spend more time on a problem when they have asked for hints.

## 5. DISCUSSION

We used data mining in two ways: first to search for mediator variables that are monotonically related to learning outcomes and thus amenable to analysis with standard tools like linear regression and path analysis, and second, to search for causal models of learning that allowed us to investigate mediation relationships and to estimate the total and indirect effects of multiple representations on learning.

Contrary to our expectations, we found that raw measures of error-rate, hint-use, and time-spent were as predictive of learning as any of the non-monotonic variants we searched over. One might suspect that our variable search failed to

[9] The paths from the interaction variable mr*se track the effect of both treatments compared to either one alone or neither. The paths from the individual treatments track the effect of each treatment when the other is absent.

improve on the apparent monotonicity of the raw measures because our sample did not include high prior knowledge students. However, students' pretest scores covered a broad range from very low to very high (see Tables I and III). Although surprising, our findings can be taken as encouraging for the community of educational data mining and for the community of researchers who study ITSs. Analyzing raw measures of error-rate, hint-use, time-spent and learning is much easier than analyzing non-monotonic variants. Furthermore, most research that uses log data obtained from ITSs assumes monotonicity. Our findings do nothing to undermine this practice.

Our findings from path analysis modeling demonstrate the importance of model search. None of our initial hypotheses fit the data, but there are millions of plausible alternatives, only a small handful of which could be practically investigated by hand. Further, estimating path parameters with a model that does not fit the data is scientifically unreliable. Parameter estimates, and the statistical inferences we make about them with standard errors etc., are all conditional on the model specified being true everywhere except the particular parameter under test.

Even if our initial hypotheses had fit the data well, however, it would have been important to know whether there were alternatives that explained the same data. The GES algorithm implemented in Tetrad IV enabled us to find plausible models that fit the data well. The models we found in Fig. 2 and Fig. 3 allow us to estimate and test path parameters free from the worry that the model within which the parameters are estimated is almost surely mis-specified, as is the case for the model in Fig. 1.

Several caveats need to be emphasized, however, lest we give the false impression that we think we have "proved" the causal relationships that appear in the path diagrams shown in Fig. 1 and Fig. 2. First, the GES algorithm assumes that there are no unmeasured confounders (hidden common causes), an assumption that is almost certainly false in this and in almost any social scientific case, but one that is routinely employed in most observational studies.[10] In future work we will apply algorithms (e.g., FCI) that do not make this assumption, and see whether our conclusions are robust against this assumption. Second, although we did include intervention interaction in our model search for experiment 1, and did test for interactions between pretest and mediators in experiment 2, by no means were our tests exhaustive, and by no means can we rely on the assumption that the true relations between the variables we modeled are linear, as the search algorithms assume. Nevertheless, many of the bivariate relationships in the data we modeled appear approximately linear, so the assumption is by no means unreasonable. Third, we have a sample of 290 students, and although that is sizable compared to many ITS studies, model search reliability goes up with sample size but down with model complexity and number of variables, and is overall impossible to put confidence bounds over on finite samples [19].

Nevertheless, our searches for causal models suggest that there are indeed path models that are consistent with our background theory and with the data, and which indicate that multiple representations enhance learning, but not through any detectible mechanism involving error-rate, hint-use, or time-spent. In experiment 1, multiple representations have a positive influence on learning, but have no detectible effect on any of the mediators we measured. In experiment 2, in which interactive graphical representations were part of the intervention, it appears that there is a mediated influence on learning through error-rate, but it is a negative influence. Research from a variety of domains shows that some interventions that decrease performance during the learning phase by increasing the variability of learning tasks result in better long-term retention and transfer performance [9,16]. In other words, interventions that are beneficial in the long run often come at some cost, for instance in the form of lower performance during the learning phase. Our results show that "costs" which become apparent during the learning phase are indeed associated with lower performance also on the posttests. However, we have not yet identified the mediators of the benefits of learning with multiple representations. Given the results from the two experiments described in the present paper, it is unlikely that the advantage of multiple representations is mediated through error-rate, hint-use, or time-spent. Taken together, the results from our two experiments suggest that researchers need to look elsewhere for the cognitive mechanisms by which multiple representations improve students' learning.

The finding that error-rate partially mediates the effect of multiple representations in experiment 2 (but not in experiment 1) is an interesting one as well. One difference between experiment 1 and experiment 2 was that the graphical representations in experiment 1 were not interactive tools, but static pictures that updated when students entered the correct answer into a text field. By contrast, the graphical representations in experiment 2 were interactive: students could drag-and-drop sections from one representation into another and use buttons to partition the representation into fewer or more sections. It is conceivable that interactive representations provide a more direct learning experience for students, which will have a different effect on problem-solving behavior (as, for example, on error-rate) than relatively static representations [18]. There is currently very little research that systematically investigates the impact of interactive versus static representations on students' problem-solving behaviors and consequent learning outcomes. Our findings

[10] Although our data are from a study in which we intervened on intervention, we did not directly intervene on our mediator or outcome variables. Thus these parts of our model are subject to the same assumptions as a non-experimental study.

demonstrate, that the impact of interactive representations is an interesting question to address in future research.

In conclusion, our results are of interest both to the educational psychology literature and to the intelligent tutoring systems literature. First, we can gain insights into the effects of instructional interventions: although multiple representations seem to overall be beneficial, they also seem to lead students to make more errors during the learning phase, which is associated with lower performance on posttests. Second, once we gain knowledge about which learning behaviors are adaptive and which are not, we can use these insights to improve our tutoring systems. For example, perhaps multi-representational ITSs should be designed to prevent errors in the practice and learning phase. Perhaps we can help students avoid practice errors by providing more worked examples, or by designing better error feedback messages. Or perhaps the increase in errors is simply a cost associated with multiple representations that instructors have to live with. These questions and others arose from path analysis and model search and lead almost directly to new hypotheses that we, and hopefully others, will address in future research.

## 6. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation, REESE-21851-1-1121307, and by the Pittsburgh Science of Learning Center which is funded by the National Science Foundation, award number SBE-0354420. In addition, we would like to thank Vincent Aleven, Nikol Rummel, Ken Koedinger, the Datashop team, and the students and teachers who participated in our study.

## 7. REFERENCES

[1] Aleven, A., Stahl, E., Schworm, S., Fischer, F., & Wallace, R. 2003. Help Seeking and Help Design in Interactive Learning Environments. Review of Educational Research, 73(3), 277-320.

[2] Ainsworth, S. 2006. A conceptual framework for considering learning with multiple representations. Learning and Instruction, 16, 183-198.

[3] Anderson, J. R. 1993. Problem Solving and Learning. American Psychologist, 48(1), 1-35.

[4] Baker, R., et al. 2008. Why Students Engage in "Gaming the System" Behavior in Interactive Learning Environments. Journal of Interactive Learning Research, 19(2), 185-224.

[5] Bollen, K. 1989. Structural Equations with Latent Variables. Wyley, New York.

[6] Charalambous, C. Y., & Pitta-Pantazi, D. 2007. Drawing on a Theoretical Model to Study Students' Understandings of Fractions. Educational Studies in Mathematics, 64, 293-316.

[7] Chickering, D. M. 2002. Optimal Structure Identification with Greedy Search. Journal of Machine Learning Research 3, 507-554.

[8] Croteau, E., Heffernan, N. T., & Koedinger, K. R. 2004. Why Are Algebra Word Problems Difficult? Using Tutorial Log Files and the Power Law of Learning to Select the Best Fitting Cognitive Model. Lecture Notes in Computer Science, 32200, 15-53.

[9] De Croock, M. B. M., Van Merrienboer, J. J. G., & Paas, F. G. W. C. 1998. High versus low contextual interference in simulation-based training of troubleshooting skills: Effects on transfer performance and invested mental effort. Computers in Human Behavior, 14(2), 249-267.

[10] Koedinger, K., & Corbett, A. 2006. Cognitive Tutors: Technology Bringing Learning Sciences to the Classroom. In: Sawyer, R. K. (ed.).The Cambridge handbook of the learning sciences, pp. 61-77, Cambridge University Press, New York, NY, US.

[11] Koedinger, K., Anderson, J. R., Hadley, W., & Mark, M. 1997. Intelligent Tutoring Goes to School in the Big City. International Journal of Artificial Intelligence in Education, 8, 30-43.

[12] National Mathematics Advisory Panel. 2008. Foundations for Success: Report of the National Mathematics Advisory Board Panel, U.S. Government Printing Office.

[13] Newell, A., & Rosenbloom, P. 1981. Mechanisms of Skill Acquisition and the Law of Practice. In Anderson, J. (ed.). Cognitive Skills and Their Acquisition, Erlbaum Hillsdale NJ.

[14] Pearl, J. 2000. Causality: Models, Reasoning, and Interference. Cambridge University Press.

[15] Rau, M. A., Aleven, V., & Rummel, N. 2009. Intelligent tutoring systems with multiple representations and self-explanation prompts support learning of fractions. International Conference on Artificial Intelligence, 441-448.

[16] Rau, M. A., Aleven, V., Tunc-Pekkan, Z., Pacilio, L., & Rummel, N. accepted. How to schedule multiple graphical representations? A classroom experiment with an intelligent tutoring system for fractions. To appear in the proceedings of ICLS 2012.

[17] Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. 2007. Cognitive Tutor: Applied Research in Mathematics Education. Psychometric Bulletin & Review, 14(2), 249-255.

[18] Robers, Y. 1999. What Is Different about Interactive Graphical Representations? Learning and Instruction, 9, 419-425.

[19] Robins, J., Scheines, R., Spirtes, P., & Wasserman, L. (2003). Uniform Consistency in Causal Inference, Biometrika, 90, 491 – 515.

[20] Spirtes, P. Glymour, C., Scheines, R. (2000). Causation, Prediction, and Search. 2nd Edition, MIT Press.

# The Impact on Individualizing Student Models on Necessary Practice Opportunities

Jung In Lee
Language Technologies Institute
Carnegie Mellon University
junginl@cs.cmu.edu

Emma Brunskill
Computer Science Department
Carnegie Mellon University
ebrun@cs.cmu.edu

## ABSTRACT

When modeling student learning, tutors that use the Knowledge Tracing framework often assume that all students have the same set of model parameters. We find that when fitting parameters to individual students, there is significant variation among the individual's parameters. We examine if this variation is important in terms of instructional decisions by computing the difference in the expected number of practice opportunities required if mastery is assessed using an individual student's own estimated model parameters, compared to the population model. In the dataset considered, we find that a significant portion of students are expected to perform twice as many practice opportunities if the student is modeled using a population-based model, compared to the number needed if the student's own model parameters were used. We also find an additional significant portion of students will be likely to receive less practice opportunities than needed, implying that such students will be advanced too early. Though further work on additional datasets is needed to explore this issue in more depth, our results suggest that considering individual variation in student parameters may have important implications for the instructional decisions made in intelligent tutoring systems that use a Knowledge Tracing model.

## 1. INTRODUCTION

Both intelligent tutoring systems and live classroom instruction often assume that student learning can be adequately represented using a single model and associated set of model parameters. For example, in this paper we will focus on Knowledge Tracing [8], a popular method for estimating student mastery of skills that has been used in effective cognitive tutor systems [9]. Knowledge tracing is parameterized by 4 variables that are typically assumed to be the same for all students. Note that these population-level models still allow us to represent variation in our estimates of student performance: if two students respond differently to a set of practice opportunities, the model will have different estimates of future student performance for the two cases.

There have been some prior work on KT student models that represent differences in the student's initial knowledge [10]. In addition, several logistic regression-based student models, including Additive Factor Models [5] and Instructional Factors Analysis [7], include a single constant that is individually fit per student. Including this student parameter has been shown to lead to models that better fit the data, and have improved prediction accuracy. However, in all these cases, the parameters related to the progress of student learning and student observations, are fitted to the entire population. Therefore the underlying dynamical process of student learning, and the way in which that is translated to student performance, is assumed to be identical across students.

There's evidence to suggest this assumption is too strong. Standard high schools commonly offer multiple versions of the same class, such as a remedial version, normal version, and honors version. This approach is taken, at least in part, because it is believed both that students may have different learning speeds or prior backgrounds for a subject, and that those differences mean that the students will be best taught in different ways. In other words, instruction will vary not just according to our current estimate of student performance, but also how we anticipate that performance changes over time.

Here we examine the variation among individual student's parameters, and quantify the impact of this variation on pedagogical strategies. To start we consider this in the context of mastery learning, using the Knowledge Tracing framework to estimate and monitor student skill mastery. We already know from Cen et al. [6] that tuning the KT parameters can lead to a significant impact on reducing the amount of necessary practice opportunities; however, this work still uses a single set of KT parameters for all students. Corbett and Andersen [8] did try fitting individual parameters, and found this improved the predictive power of the model, as well as some evidence that this might improve student performance; however, the authors used curve-fitting to find the parameter values[1] and the authors did not examine the difference in practice opportunities needed if a population model was used instead of an individual model.

In this paper we fit Knowledge Tracing model parameters to each individual, on a dataset from the ASSISTment system [10]. We examine the distribution of the resulting parameters, and compare them to computing a single set of KT model parameters for all students. In our second contribution, we compute the difference in expected number of practice opportunities required if mastery is assessed using an individual student's own estimated model parameters, compared to the population model. We find that about 40% of

[1]Evidence [2] suggests that EM, which we use in this paper, finds a better parameter fit than curve fitting.

our student sample falls into one of two cases: students who will be forced to do many more practice opportunities than necessary, and students who are considered to have mastered the material when in reality they likely need additional practice opportunities. This implies that observed variation in student modeling parameters has an important and significant effect on instructional decisions, and that considering individual variation in the model parameters could lead to more effective teaching. We will outline several future ideas for how this could be accomplished in the conclusion of this paper.

## 2. BACKGROUND

A popular approach is to model a student's knowledge as a latent variable, which changes in response to practice opportunities. The system gets information about this underlying state through the student's responses to practice opportunities. This model is a Hidden Markov model, and estimating the student's current hidden state can be done by performing Bayesian filtering. A special case of this approach is known as Knowledge Tracing [8], which assumes the student has 1 binary hidden state per skill (the student has either mastered or not mastered the skill) and binary observations, corresponding to whether the student gets a question about a skill correct or incorrect. There are 4 model parameters per skill in the Knowledge Tracing framework: $p(L_0)$ is the initial probability the student has already mastered the skill; $p(T)$ is the probability of the student transitioning from not having mastered the skill to having mastered it after a practice opportunity; $p(S)$ is the probability the student gives a wrong answer even though she has mastered the skill; and $p(G)$ is the probability the student gives the correct answer even though she has not mastered the skill. After the student is given a practice opportunity, and gets the problem correct or incorrect, the model updates the probability of the student's underlying mastery state. Though in Knowledge Tracing the standard approach is to first update the estimate of the student's mastery given their observed response, and then update their mastery as to whether they have learned, in this paper we instead adopt the alternate convention (often used in other Bayesian models) of first updating the probability the student has mastered the material given they received a new practice opportunity, and then updating that estimate given the observed student response. This yields the following equations for computing $p(L_{t+1})$, the probability the student has mastered the skill at time step $t+1$, as a function of the probability the student has mastered the skill at the prior time step $p(L_t)$, the observed student response, and the 4 parameters:

$$p(L_{t+1}|c) = \frac{(1-p(S))(p(L_t)+p(T)(1-p(L_t)))}{(1-p(S))(p(L_t)+p(T)(1-p(L_t)))+p(G)(1-p(T))(1-p(L_t))} \quad (1)$$

$$p(L_{t+1}|w) = \frac{p(S)(p(L_t)+p(T)(1-p(L_t)))}{p(S)(p(L_t)+p(T)(1-p(L_t)))+(1-p(G))(1-p(T))(1-p(L_t))} \quad (2)$$

The basic model assumes that the student never forgets a skill, so once it is mastered, it stays mastered.

Often Knowledge Tracing is paired with mastery learning. Here the goal is for the student to master the desired skill (or set of skills). As the student's mastery level is hidden, a typical approach is to continue to provide practice opportunities to the student until, based on the student's responses, the probability that the student has mastered the skill $p(L_t)$ exceeds some prespecified threshold, such as 95%.

## 3. METHODS

Our interest is in characterizing the distribution of model parameters associated with individual students, and examining how this impacts the amount of practice opportunities needed for a student to reach mastery, versus using parameters fit to the population.

### 3.1 Parameter Fitting

We fit a KT model for each individual student, where the model consists of the 4 parameters specified in the prior section, $\langle p(L_0), p(T), p(S), p(G)\rangle$. The input data for each student consists of a trajectory of practice opportunities for a particular skill, where the $j$-th entry contains whether the student got this opportunity correct or incorrect. We would like to compute the model parameters that maximize the likelihood of the observed data. If at each $j$ we knew if the student had mastered or not mastered the skill, then computing the best parameters would simply involve counting. For example, to estimate the probability of slipping $p(S)$ we would simply count up the number of instances where the student had mastered the skill but got the problem wrong, divided by the number of times the student had mastered the skill. However, we don't know if the student has mastered the skill or not. Therefore we use Expectation Maximization to find parameters that locally maximize the likelihood of the observed data.

Expectation maximization (EM) is an iterative algorithm where each iteration consists of two stages. In the first stage we fix the current estimates of the parameters and use these parameters to estimate the probability the student has mastered or not mastered ($\hat{p}(L_t)$ and $1-\hat{p}(L_t)$) the skill at each of the time steps in the trajectory. These estimates can be efficiently computed using the forward-backward algorithm, whose computational complexity is linear in the trajectory length and quadratic in the number of hidden states: here there are only 2 possible states, mastered or unmastered.

In the second stage, new parameter estimates are computed given these estimated probabilities of mastery. The new $p(S)$ parameter is computed by taking all instances where the student got the problem wrong, and summing the probability that the student had really mastered the skill in all those instances, divided by the probability of the student having mastered the skill on all time steps:

$$\hat{p}(S) = \frac{\sum_j \delta_j(w)\hat{p}(L_j)}{\sum_j \hat{p}(L_j)},$$

where $\delta_j(wrong)$ is 1 if the $j$-th student response was wrong, and 0 otherwise. A new estimate of $p(G)$ is computed as

$$\hat{p}(G) = \frac{\sum_j \delta_j(c)(1-\hat{p}(L_j))}{\sum_j (1-\hat{p}(L_j))},$$

where in the numerator we sum over all instances where the student got the answer correct. Updating the parameter estimate of $p(T)$ involves the probability of the student transitioning from not having the skill mastered to having mastered the skill. $p(L_0)$ is estimated from $\hat{p}(L_0)$.

**Algorithm 1** ExpOppNeed

**Input:** $p^{tsg} = \langle p(T), p(S), p(G) \rangle$, $p(L_j)$, $d$, $p_{path}$, $\epsilon$
**Output:** $EO$
**if** $p_{path} < \epsilon$ **then**
  return $EO = 0$;
**else**
  **if** $p(L_j) \geq d$ **then**
    return $EO = 0$ {reached mastery}
  **else**
    $p(c|p(L_j))$ (Eqn. 3) {prob get correct}
    $p(L_{j+1}|c)$ (Eqn. 1) {prob mastery if get correct}
    $p_{path,c} = p_{path} * p(c|p(L_j))$
    {Compute further opp. need if get problem correct}
    $EO_c$=ExpOppNeed($p^{tsg}$, $p(L_{j+1}|c)$, $d$, $p_{path,c}$, $\epsilon$)
    $p(w|p(L_j))$ (Eqn. 4) {prob get wrong}
    $p(L_{j+1}|w)$ (Eqn. 2) {prob mastery if get wrong}
    $p_{path,w} = p_{path} * p(w|p(L_j))$
    {Compute further opp. need if get problem wrong}
    $EO_w$=ExpOppNeed($p^{tsg}$, $p(L_{j+1}|w)$, $d$, $p_{path,w}$, $\epsilon$)
    return $EO = 1 + p(c|p(L_j)) * EO_c + p(w|p(L_j)) * EO_w$
  **end if**
**end if**

Then the whole process repeats, using the updated parameter estimates to compute new estimates of the underlying probability of mastery. These iterations continue until the process converges to a fixed point, which is guaranteed to occur.

There are several limitations to using EM that are relevant for our purposes. First, EM is only guaranteed to converge to a local optima of the likelihood function. Second, the parameters found may not be semantically plausible. This can occur for the $p(S)$ and $p(G)$ parameters. We expect $p(S)$ to be $< 0.5$ since if it is larger, it means that it is more likely for a student to get a problem wrong than right when she has mastered the given skill. Similarly, we expect $p(G)$ to be $< 0.5$, since a student is more likely to get a problem wrong than right if she has not mastered the associated skill.

Given these concerns, we performed a discretized search over the $p(G)$ and $p(S)$ parameters, as well as a discretized search over the initial probability of mastery $p(L_0)$. We included this parameter as we initially had some results where the $p(L_0)$ was fit by EM to be 0 or 1 and we suspect it is probable that students should lie inside these extremes. We ran EM to compute the best $p(T)$ for each tuple of $\langle p(L_0), p(G), p(S) \rangle$ parameters, and selected the model parameter tuple with the highest likelihood.

In addition, we also fit model parameters by aggregating all student data together, and fitting a single population model. In the rest of the paper we will use $p_i$ to denote the parameters fit for the $i$-th student, and $p_{pop}$ to denote the parameters fit for the whole population of students.

## 3.2 Expected Time to Mastery

Given the estimated student learning model parameters, we next compute the expected number of practice opportunities it will take for student $i$ to reach mastery. We take the standard approach of using a threshold $d$ to define mastery: when $p(L_t) \geq d$, the student is defined to have reached mastery. Consider a given student $i$, with her associated parameters $\langle p_i(L_0), p_i(T), p_i(S), p_i(G) \rangle$. First, let $p_i(L_t|obs_{1:t})$ be the probability of student $i$ having mastered the skill after having $t$ practice opportunities, and having made the responses $\langle obs_1, obs_2, \ldots obs_t \rangle$ to each respective practice opportunity (where $obs_j = correct, wrong$). Note this expression can be calculated by sequentially applying either Equation 1 or Equation 2, depending on whether the student got that practice opportunity correct or incorrect.

**Algorithm 2** ExpOppNeedPop

**Input:** $p_i^{tsg} = \langle p_i(T), p_i(S), p_i(G) \rangle$, $p_i(L_j)$,$d$,$p_{path}$, $\epsilon$, $p_{pop}^{tsg} = \langle p_{pop}(T), p_{pop}(S), p_{pop}(G) \rangle$, $p_{pop}(L_j)$,
**Output:** $EO$
**if** $p_{path} < \epsilon$ **then**
  return $EO = 0$;
**else**
  **if** $p_{pop}(L_j) \geq d$ **then**
    return $EO = 0$ {reached mastery under pop model}
  **else**
    $p(c|p_i(L_j))$ (Eqn. 3) {prob student $i$ gets correct}
    {Prob. mastery if get correct}
    $p_{pop}(L_{j+1}|c)$ (Eqn. 1) {under population model}
    $p_i(L_{j+1}|c)$ (Eqn. 1) {under student $i$'s model}
    $p_{path,c} = p_{path} * p(c|p_i(L_j))$
    {Compute further opp. need if get problem correct}
    $EO_c$=ExpOppNeedPop($p_i^{tsg}$, $p_i(L_{j+1}|c)$, $d$, $p_{path,c}$, $\epsilon$, $p_{pop}^{tsg}$,$p_{pop}(L_{j+1}|c)$)
    $p(w|p(L_j))$ (Eqn. 4) {prob student $i$ gets wrong}
    {Prob. mastery if get wrong}
    $p_{pop}(L_{j+1}|w)$ (Eqn. 2) {under population model}
    $p_i(L_{j+1}|w)$ (Eqn. 2) {under student $i$'s model}
    $p_{path,w} = p_{path} * p(w|p(L_j))$
    {Compute further opp. need if get problem wrong}
    $EO_w$=ExpOppNeedPop($p_i^{tsg}$, $p_i(L_{j+1}|c)$, $d$, $p_{path,c}$, $\epsilon$, $p_{pop}^{tsg}$, $p_{pop}(L_{j+1}|c)$)
    return $EO = 1 + p(c|p_i(L_j)) * EO_c + p(w|p_i(L_j)) * EO_w$
  **end if**
**end if**

A second important pair of quantities is the probability of observing a particular student response (correct or incorrect) at time $t$ to an opportunity, given the current probability of mastery $p_i(L_t)$. These are:

$$p(c|p_i(L_t)) = p(G)(1 - p_i(L_t)) + (1 - p(S))p_i(L_t) \quad (3)$$

$$p(w|p_i(L_t)) = (1 - p(G))(1 - p_i(L_t)) + p(S)p_i(L_t). \quad (4)$$

Using Equations 1,2,3 and 4, we can compute the expected number of practice opportunities for a student to reach mastery using a recursive algorithm. Intuitively, the expected number of additional practice opportunities needed depends on the current probability that the student has mastered the material, $p_i(L_t)$. If $p_i(L_t) \geq d$ then no more practice opportunities are needed. Otherwise at least one more practice opportunity is needed. Depending on whether the student gets that opportunity correct or incorrect, then $p_i(L_{t+1})$ will get updated accordingly, and then we can compute the expected number of additional opportunities needed from the resulting probability of mastery. But we don't know in advance whether the student will get the next question correct or not, so we take the expectation over these two possibilities. More precisely, let $EO_i(p_i(L_t))$ be the expected number of further practice opportunities needed given the

current probability of mastery. Then if $p_i(L_t) \geq d$,

$$EO_i(p_i(L_t)) = 0, \quad (5)$$

otherwise,

$$\begin{aligned} EO_i(p_i(L_t)) &= 1 + p(c|p_i(L_t))EO_i(p_i(L_{t+1}|c)) \\ &+ p(w|p_i(L_t))EO_i(p_i(L_{t+1}|w)). \end{aligned} \quad (6)$$

The full algorithm, ExpOppNeed, is displayed in Algorithm 1. The algorithm is called using the student's initial probability of mastery, $p_i(L_0)$. Calculating the expected number of practice opportunities needed for individual $i$ is done using Equations 5 and 6. The algorithm also maintains a variable $p_{path}$ that represents the probability of the path of observations, given the student's initial probability of mastery $p_i(L_0)$. When $p_{path}$ falls below a threshold $\epsilon$, we also set the expected number of future practice opportunities needed to 0. This is an approximation, but is necessary since in some models the student will never reach a sufficient probability of mastery if she gets all practice opportunities incorrect. Since in the real world such a student would not do an infinite number of problems, and to prevent infinite recursion, we terminate when $p_{path} < \epsilon$.

## 3.3 Population vs Individual

As mentioned previously, often intelligent tutoring systems operate with a fixed set of parameters for all students, which we denote the population model $p_{pop}$. We are interested in what impact this assumption means for individual students whose learning parameters may not match the aggregate population model parameters. More specifically, we are interested in the expected needed number of practice opportunities if the probability used to assess mastery is calculated using the population parameters instead of the student's own parameters. Note that we cannot simply apply Algorithm 1 using the population parameters. Doing so would calculate the expected number of practice opportunities needed if we assess mastery using the probability given by the population model, and if correct and wrong observations are generated according to a student who has the same parameters as the population model. In contrast, we are interested in the case when a real student (with different parameters than the population model) is responding correctly or incorrectly to practice opportunities, but that the system is monitoring the student's progress using a different set of model parameters. This is likely to occur in tutors that use population models.

To estimate the expected needed number of practice opportunities in this situation, we need to estimate the probability of a correct or wrong answer being generated according to student $i$'s model parameters $p_i$, but assess if the student has reached a sufficient probability of mastery using the population parameters $p_{pop}$. This means that during our calculations we need to maintain two separate estimates of the probability of mastery for the student, $p_{pop}(L_t|obs_{1:t})$ and $p_i(L_t|obs_{1:t})$, which respectively represent the probability under the population model parameters, given the observed sequence of correct and incorrect answers, and the probability under student $i$'s own parameters.

Let $EO_i^{pop}(p_{pop}(L_t), p_i(L_t))$ represent the expected number of additional practice opportunities need to reach mastery, given the current estimate of the probability of mastery under the population model is $p_{pop}(L_t)$, and under the individual model is $p_i(L_t)$. $EO_i^{pop}$ is computed in Algorithm 2, which is a modification of Algorithm 1.

# 4. EXPERIMENTS

We used a dataset of student responses to 42 problem sets in the ASSISTment system [10]. Each set corresponded to 1 skill, and the number of problems given per skill ranged from 4-13. We first compute individual student parameters. Since the number of data samples was fairly limited per skill, we computed a set of parameters for all skills an individual did problems on. Though this is an approximation, in existing tutoring systems with good learning outcomes, multiple skills are often modeled as having the same parameters. We will consider this and other assumptions made further in the discussion section. We selected the subset of 265 students who did problems on 10 or more skills. For this subset, the mean number of skills per student was 12.79 (range=[10,22]), yielding an average of 69.57 (range=[45,132]) total problem tries per student. We fit learning parameters $\langle p(L_0), p(T), p(S), P(G)\rangle$ to each individual, and to the aggregated dataset across all individuals.

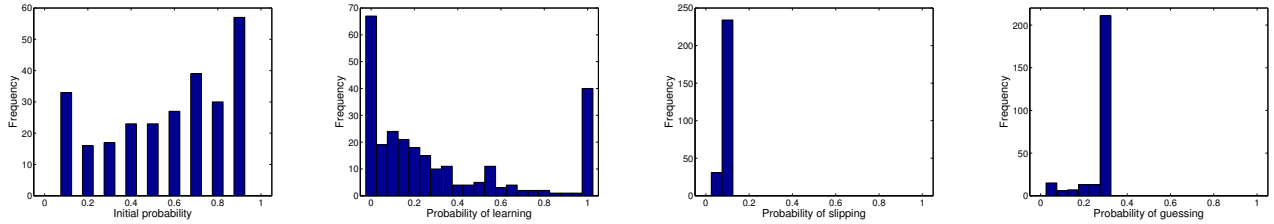
When fitting the model parameters, we restricted $p(S)$ to lie in $[0.05, 0.1]$ and $p(G)$ to lie in $[0.05, 0.3]$, trying values incremented by 0.05: these bounds have been used in prior work [1]. $p(L_0)$ was restricted to lie in $[0.1, 0.9]$, and values were tried at increments of 0.1. $p(T)$ was fit using EM, as described previously. We ran EM for each individual (or the aggregated dataset) with 10 different initializations, and chose the one with the highest log likelihood. The computational time needed for EM to converge (for a single individual's data) was approximately 0.6s.

Following the standard procedure in Knowledge Tracing [8], we set the mastery threshold $d$ to 0.95. We set the expected number of future practice opportunities from $p(L_t|obs_{1:t})$ to 0 if the sequence of observations $obs_{1:t}$ had a probability of less than $\epsilon = 10^{-7}$. We later discuss the effect of $\epsilon$.

# 5. RESULTS

We first report the resulting distributions of the estimated student learning model parameters, computed for each student separately. Histograms of these distributions are displayed for the four learning parameters in Figures 1(a), 1(b), 1(c), and 1(d). For the slip $p(S)$ and guess $p(G)$ parameters, the distribution is fairly peaked. The majority of individuals have $p(S)$ and $p(G)$ parameters that are close to the parameters for the full population. However, for both the initial probability of mastery, $p(L_0)$, and the probability of mastering a skill after not having understood it, $p(T)$, there is a large spread of values. In each figure we have also included the parameter estimated if the data from all 265 individuals is aggregated, and a single set of model parameters is estimated. It is clear that for both $p(L_0)$ and $p(T)$, there will be many individuals whose best fit parameters are quite far from the population parameters. This suggests that the expected number of practice opportunities needed for some students may differ if student mastery modeling is done using the individual student's own estimated parameters, compared to if we maintain a probability of the student's mastery level using the population parameters.

Indeed, this is what we observe. We start by computing the expected number of practice opportunities needed for




(a) $p(L_0)$. $p_{pop}(L_0) = 0.6$ (b) $P(T)$. $p_{pop}(T) = 0.086$ (c) $P(S)$. $p_{pop}(S) = 0.1$ (d) $P(G)$. $p_{pop}(G) = 0.3$

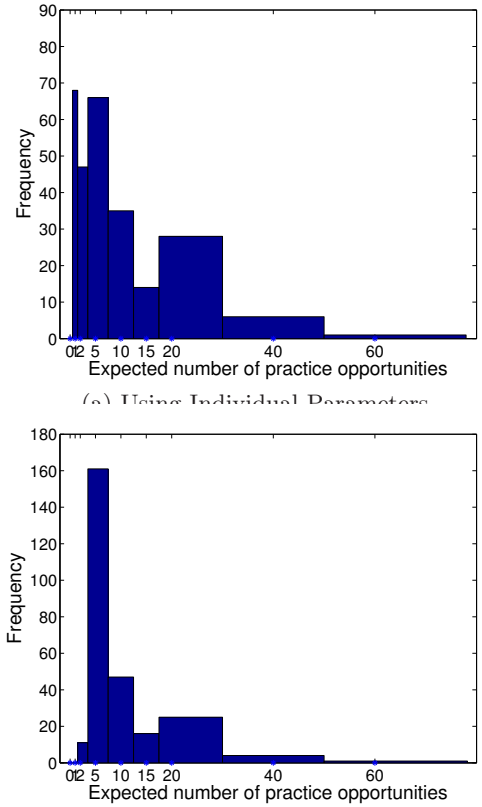
Figure 1: Histogram of parameters fit for each individual.

an individual, given the model parameters $p_i$ estimated for that individual, $EO_i(p_i(L_0))$. The probability of the student reaching mastery is assessed using the student's parameters $p_i$. A histogram of the expected number of practice opportunities for an individual to reach mastery as evaluated using their own parameters is displayed in Figure 2(a). We have binned the data to illustrate the distribution of the individuals. Over half the individuals are expected to need less than 5 practice opportunities to reach mastery. However, there are also a significant number of individuals that are expected to need a much larger number of practice opportunities to reach mastery: over 40 students require at least 15 practice opportunities, and over 30 of those require 20 or more. This graph suggests that there is a significant spread in the required amount of practice necessary for different students.

We next examine the expected number of practice opportunities needed for a student $i$, if we evaluate mastery using the population parameters, $EO_i^{pop}(p_{pop}(L_0), p_i(L_0))$. Note that this is the situation that occurs in existing tutors that use a single set of model parameters for all students. Figure 2(b) displays a histogram of the expected number of practice opportunities needed when evaluating mastery using the population parameters. At a rough glance the histogram looks similar to Figure 2(a), but upon closer examination, the individuals histogram has its peak at about 2 whereas the population model has its peak at about 5 expected number of time steps.

We examine this discrepancy more systematically in Figures 3(a) and 3(b). There are three possible situations that could result from estimating the number of practice opportunities if we evaluate mastery learning using a set of population parameters, on an individual with their own set of parameters. In the first case a student might have to do more practice problems than they would if we evaluated mastery using the student's own parameters. For example, this might occur if the probability of mastering an unmastered skill is higher for individual $i$ than for the group, $p_i(T) > p_{pop}(T)$. In the second case an individual will need a very similar number of practice opportunities, whether we evaluate their skill mastery using the estimated population parameters, or the individual's own parameters. In the third case are students who are expected to need more practice opportunities than that predicted if using the population model.

The first and third situations are the ones of concern. In the first case, it would mean that some students are having to do more problems than really needed. Since there is only a finite amount of time in the school year, this means that these students would be likely to cover less material than they are




(a) Using Individual Parameters

(b) Using Population Parameters

Figure 2: The expected number of practice opportunities to reach mastery, where the probability of mastery is evaluated either based on the individual's estimated model parameters, or the estimated population model parameters.

capable of, due to doing extra unnecessary problems. In addition, doing more problems than required might contribute to student boredom and disengagement. In the third case, a student may be recorded as having mastered a skill before he has really reached the required threshold probability of mastery. This means that the student could be advanced to later skills, some of which might assume knowledge of this earlier skill, without the student having actually understood the earlier skill to a satisfactory level.

Therefore our next objective was to ascertain if there was evidence that either case 1 or case 3 occurred in this dataset. For each individual $i$ we computed the difference in expected number of practice opportunities needed using the popula-

tion model to assess mastery, versus the individual model:

$$\Delta(i, pop) = EO_i^{pop}(p_{pop}(L_0), p_i(L_0)) - EO_i(p_i(L_0))). \quad (7)$$

Figure 3(a) displays a histogram of $\Delta(i, pop)$. Note that higher values mean that the expected number of practice opportunities is more under the population parameters, than if one used the individual's parameters, aka case 1. Negative numbers are when the individual is expected to need more opportunities given their own parameters, compared to the population parameters, aka case 3. We can see that there are a number of individuals that fall into either case. One way for defining students in case 1 is students that are expected to be given one or more additional practice opportunities if the population model is used to do assessment of reaching the threshold probability of mastery, versus the expected number using the student's own parameter. Using this definition 146 students (or 55%) fall into case 1, where the mean $\Delta(i, pop)$ of this group is 2.18. Though important, this may not sound like a large number. But the importance of this difference becomes more clear when we consider the ratio of expected practice opportunities,

$$r(i, pop) = \frac{EO_i^{pop}(p_{pop}(L_0), p_i(L_0))}{EO_i(p_i(L_0))}. \quad (8)$$

A histogram of $r(i, pop)$ is displayed in Figure 3(b). This figure shows that over 50 students in our dataset would be expected to do *twice* as many problems if the population parameters are used to assess mastery learning, compared to using the individual's parameters. For example, one student's parameters were $p_i(L_0) = 0.3, p_i(T) = 0.86, p_i(S) = 0.1, p_i(G) = 0.3$. Here $\Delta_i(i, pop) = 2.74$ and $r(i, pop) = 3.32$. For such students, they could potentially be covering twice or more as much material in the same time if their own model parameters were known and used to evaluate mastery. This could have very significant effects on such students' learning, and, equally importantly, at least in this dataset, this effects a significant proportion of the students, over 20% of the student population in our dataset.

Returning to estimates of $\Delta(i, pop)$, Figure 3(a) also reveals that there are a number of individuals whose $\Delta(i, pop) < -1$, indicating that we expect such individuals to need at least one more practice opportunity if we use their own parameters to assess mastery, compared to using the population model parameters. This is case 3, and there are 21 individuals who fall into this category. The average $\Delta(i, pop)$ for this subset is $-1.47$. An additional interesting quantity in this case is the expected probability of mastery of individual $i$, if mastery is assessed using the population model parameters. This quantity is displayed in Figure 3(c) for all students whose $\Delta(i, pop) < 0$, aka students that are expected to need more practice opportunities than they will receive if the population parameters are used. Though a subset of these students have an expected probability of mastery that is fairly close to the desired threshold (0.95), 44 students ($\sim$17%) have an expected probability of mastery of 0.6 or less. For example, one student's estimated parameters were $p_i(L_0) = 0.1, p_i(T) = 0.028, p_i(S) = 0.1, p_i(G) = 0.3$, and the expected actual average mastery for this student when his mastery is assessed using the population parameters, was only 0.47, far below the 0.95 threshold of mastery desired. Such individuals are unlikely to have sufficiently understood the current skill, yet would be considered to have reached mastery, and moved on to the next skill.

Therefore, in this dataset, there appear to be a substantial number of students who fall into case 1 or case 3, who would be likely to have to do more problems than is actually necessary, or who would not have reached the desired probability of mastery, respectively.

Some readers might be concerned that the reported results are computed using an algorithm that only approximates the expected number of needed practice opportunities. This approximation arises from setting $EO_i(p(L_t|obs_{1:t})$ to 0 if the probability of the path of observations is less than set threshold, $p(obs_{1:t}) \leq \epsilon = 10^{-7}$, which was done from computational reasons. As one exploration, we repeated our analysis using $\epsilon = 10^{-5}$ and observed very similar results. More generally, our approximation will typically underestimate the difference $\Delta(i, pop)$ in the two cases we are interested in. Intuitively this is because whichever parameter set, either $p_i$ or $p_{pop}$, has the higher learning rate and/or higher initial probability of mastery, that model will be less effected by terminating low probability paths, because its expected number of time steps will be shorter. We tested this intuition by taking the population parameters (where $p(T) = 0.086$), and creating two alternative models, one "fast" learner where $p(T) = 0.2$, and one "slow" learner where $p(T) = 0.02$. We used different $\epsilon$s and evaluated $\Delta$ for these different models:

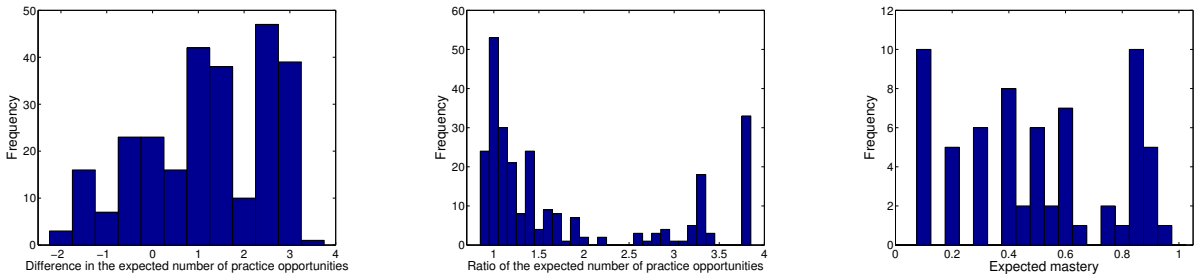
| $\epsilon$ | $10^{-5}$ | $10^{-6}$ | $10^{-7}$ | $0.5 * 10^{-7}$ |
|---|---|---|---|---|
| $\Delta(fast, pop)$ | 2.511 | 2.844 | 3.103 | 3.175 |
| $\Delta(slow, pop)$ | $-1.700$ | $-2.374$ | $-3.100$ | $-3.311$ |

Note that the magnitude of the difference is increasing as $\epsilon \to 0$, indicating that by using a $\epsilon > 0$, we are likely to be underestimating the true difference in expected time steps between an individual model and when using a population model. This suggest that we expect our reported results to be an underestimate of the true significance of the impact of using population parameters instead of individual parameters when assessing mastery.

## 6. RELATED WORK

Student modeling is naturally of key interest to the intelligent tutoring systems and educational data mining community. Knowledge Tracing [8] has been explored extensively in the research community and is also used in effective intelligent tutoring systems.

Over the last 5 years there has been significant interest in methods for fitting the parameters in the Knowledge Tracing model. Beck and Chang [3] pointed out that in some cases, more than 1 set of KT parameters predict exactly the same student performance (the probability the student will get the next answer correct as the number of practice opportunities increases). This means that the model suffers from an identifiability problem, where there are more than 1 set of parameters that equally well fit the observed data. In addition, Beck and Chang discuss the issue that when model parameters are fit by EM, the resulting model parameters may be implausible from a seminar perspective, for example if the probability of guessing the right answer $p(G)$ is higher than 0.5, then students that have a greater chance of getting a problem right than wrong when they haven't mastered the skill, which seems unlikely. Beck and Chang addressed the issue of identifiability by using hand set Dirichlet priors to introduce domain. Baker, Corbett and Aleven [1] presented a machine learning method to es-




(a) Population-Individual (b) Ratio $\frac{Population}{Individual}$ (c) Expected mastery

Figure 3: The expected number of practice opportunities needed when using the population parameters to assess if the student has reached a threshold of mastery(a), versus the individual's own parameters(b). (c) shows the expected level of mastery of students if using the population parameters to assess mastery, for students $i$ that needed more practice, $\Delta(i, pop) > 0$.

timate the probability a student has slipped or guessed as a way to address both identifiability and plausibility. Rai and colleagues [11; 12] have since investigated learning Dirichlet priors from data, or multiple Dirichlets, to improve parameter plausibility. Ritter et al. [13] clustered similar skills together to reduce the number of model parameters required in order to find better parameters given a fixed set of data.

There has also been a limited amount of work on trying to individualize the Knowledge Tracing parameters. In their key paper, Corbett and Andersen [8] investigated individualizing the KT parameters by first learning a set of parameters for all students, and then computed weights to adjust these parameters to individual students. While the resulting weighted parameters lead to predictions that correlated better on student test performance than a nonweighted model, the weighted model did not generally have a better predictive accuracy on the student's test score. In contrast to weighting, Pardos and Heffernan [10] simply enabled the initial probability of mastery, $p(L_0)$, to vary among individuals during the initial fitting process. The resulting model predicted student responses better than a Knowledge Tracing model where all parameters are identical for all students. Our work is related to Pardos and Heffernan, except we fit all 4 KT parameters individually to each student.

There are a number of alternate mathematical models of student learning and performance, including Learning Factors Analysis [5]. A number of these models use a logistic regression approach to directly modeling the probability a student will get the next practice opportunity correct, without an additional representation of the student's latent skill mastery state. However to our knowledge, unlike the Knowledge Tracing approach, these models have not been used to help make decisions about how much additional practice a student needs. One simple extension to the idea of thresholding when the student's probability of mastery has reached a sufficient level would be to continue providing the student with practice opportunities until the logistic regression model exceeds a prefixed threshold of the student getting the next problem correct.

## 7. DISCUSSION AND FUTURE WORK

Our results suggest there is a large amount of variation in student learning parameters in this dataset, and this variation has important implications for the amount of practice opportunities that should be given to individual students.

There are a number of ways that the work presented in this paper could be further improved in the future. Right now we are combining a brute force discretized grid search over a subset of the Knowledge Tracing parameters, with performing Expectation Maximization over the remaining parameters. It would be interesting to explore other methods for fitting KT parameters to data, such as those previously proposed [3; 1; 11; 12; 13].

We currently assume all the skill parameters for an individual student are the same. This was done in order to have a larger number of data points to use in order to fit the student model parameters, and because there is prior precedent in successful tutors of using the same skill parameters for multiple skills. However, we suspect that at least for some skills, the parameters differ. Therefore we have two axes of potential variation: the individual skills, and the individual students. One natural concern is that increasing the number of model parameters (by modeling individual skills or individual students) leads to a danger of overfitting, since there will be less data for each subset of parameters we wish to estimate. Indeed, prior work by Ritter et al. [13] found that by clustering skills into groups, and fitting KT parameters to each cluster (instead of each skill), the resulting clusters generalized better to new students than fitting data to individual skills. We plan to perform a related analysis on seeing if there appear to be different clusters of students, all of which share similar skill parameters. More generally, we are interested in developing hierarchical models of skill parameters, which could fit individual student and individual skill parameters, but do so in a way that encourages clustering of both skills and students. Latent Dirichlet Allocation [4], a method for modeling collections that share features, such as documents sharing subsets of topics, may be a relevant approach.

Though our results suggest that individual variation in student model parameters can exist, and lead to overpractice or underpractice if these differences are ignored by using a population model, we have only analyzed a single dataset. Further experiments should be performed on other datasets to see if similar results are obtained.

Assuming these results hold in other cases, to us the most exciting issue is how to modify automated mastery teaching to enhance student learning, assuming that individual students' parameters vary. In the current paper we fit individ-

ual model parameters to the student data after the students have already completed all practice opportunities. In real tutoring settings, the student parameters will be unknown. Therefore, in order to test whether using individual student parameters benefits students using real classroom studies, we will need to infer students' parameters while simultaneously monitoring their performance and deciding whether the student has reached a sufficient threshold probability of mastery. Corbett and Andersen [8] used a fixed initial set of exercises as a diagnostic period to learn student parameter weights. After this mastery learning proceeds using parameters that incorporate the student's individual weights. Of course, this begs the question of how long a diagnostic period is required to fit a good estimate of the individual's parameters, while simultaneously being short enough that the individual may benefit from having individualized parameters: if we can only fit individual parameters after the student has reached mastery, this method is unlikely to be effective. The success of this approach also depends on other modeling assumptions. For example, if we assume that an individual has the same set of parameters for all skills, then one approach is to use the population skill parameters when the student is learning the first skill, and then take the student data from that first skill and fit a set of KT parameters to that data for the individual. This set could then be used as the student learns other skills. One could also imagine taking a Bayesian approach and explicitly modeling the uncertainty over a student's learning parameters. This uncertainty could be updated as the student responds to practice opportunities. One challenge in this setting is that in a Knowledge Tracing model the student's true state of mastery is never observed. This poses some interesting technical challenges when updating a distribution over model parameters.

Finally, in this paper we focused on a Knowledge Tracing model of student learning. KT models have been often used in mastery tutoring systems. In contrast, we are not aware of prior work that use logistic regression student learning models (e.g. [5]) as part of an adaptive instructional strategy that decides when to stop giving the student practice opportunities. Logistic regression student models often incorporate both student-specific and shared population parameters. Therefore in the future we are interested in incorporating logistic regression models into a tutoring strategy, and comparing this with our individualized KT model mastery-learning approach, in terms of their effect on the expected amount of practice needed.

## 8. CONCLUSIONS

In this paper we fit all 4 Knowledge Tracing parameters to each individual student from a set of tutoring data, and examined the resulting parameter distributions. The resulting observed parameter distribution was found to have interesting implications for instruction versus using a single model for all students. About 20% of students would have to do approximately double the number of practice problems in order to reach the threshold of mastery defined using the population parameters, compared to using the student's own parameters to assess probability of mastery. Another $\sim$17% of students would be expected to have a probability of mastery of only 60% or less when the population model would expect the student is at a probability of mastery of 95% or higher. This suggests that using a single set of population parameters for all students in a tutoring system may result in a significant portion of students covering less content than they are capable of, due to being required to complete redundant practice problems, and equally importantly, may advance some students to later skills before they are ready. In the future we will investigate how to learn and incorporate individualized parameters during the tutoring process.

## 9. ACKNOWLEDGEMENTS

The authors thank the Pittsburgh Science of Learning Center and Google for support.

## 10. REFERENCES

[1] S. J. d. Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian estimation. In *ITS*, 2008.

[2] J. Beck. Difficulties in inferring student knowledge from observations (and why you should care). In *AIED Workshop on EDM*, 2007.

[3] J. Beck and K. Chang. Identifiability: A fundamental problem of student modeling. In *UM*, 2007.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[5] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis - a general method for cognitive model evaluation and improvement. In *ITS*, 2006.

[6] H. Cen, K. Koedinger, and B. Junker. Is Over Practice Necessary? Improving Learning Efficiency with the Cognitiive Tutor using Educational Data Mining. In *AIED*, 2007.

[7] M. Chi, K. Koedinger, G. Gordon, P. Jordan, and K. VanLehn. Instructional factors analysis: A cognitive model for multiple instructional interventions. In *EDM*, 2011.

[8] A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278, 1995.

[9] K. R. Koedinger, J. Anderson, W. Hadley, and M. Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43, 1997.

[10] Z. Pardos and N. Heffernan. Modeling individualization in a Bayesian networks implementation of knowledge tracing. In *UMAP*, 2010.

[11] D. Rai, Y. Gong, and J. Beck. Using Dirichlet priors to improve model parameter plausibility. In *EDM*, 2009.

[12] D. Rai, Y. Gong, and N. Heffernan. Using multiple Dirichlet distributions to improve model parameter plausibility. In *EDM*, 2010.

[13] S. Ritter, T. K. Harris, T. Nixon, D. Dickison, R. C. Murray, and B. Towle. Reducing the knowledge tracing space. In *EDM*, 2009.

# Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra

Ryan S.J.d. Baker
Worcester Polytechnic Institute
Worcester, MA USA
rsbaker@wpi.edu

Sujith M. Gowda
Worcester Polytechnic Institute
Worcester, MA USA
sujithmg@wpi.edu

Michael Wixon
Worcester Polytechnic Institute
Worcester, MA USA
mwixon@wpi.edu

Jessica Kalka
Carnegie Mellon University
Pittsburgh, PA USA
jessi.kalka@gmail.com

Angela Z. Wagner
Carnegie Mellon University
Pittsburgh, PA USA
awagner@cmu.edu

Aatish Salvi
Worcester Polytechnic Institute
Worcester, MA USA
aatishsalvi@gmail.com

Vincent Aleven
Carnegie Mellon University
Pittsburgh, PA USA
aleven@cs.cmu.edu

Gail W. Kusbit
Carnegie Mellon University
Pittsburgh, PA USA
gkusbit@cs.cmu.edu

Jaclyn Ocumpaugh
Worcester Polytechnic Institute
Worcester, MA USA
jocumpaugh@wpi.edu

Lisa Rossi
Worcester Polytechnic Institute
Worcester, MA USA
lrossi@wpi.edu

## ABSTRACT

In recent years, the usefulness of affect detection for educational software has become clear. Accurate detection of student affect can support a wide range of interventions with the potential to improve student affect, increase engagement, and improve learning. In addition, accurate detection of student affect could play an essential role in research attempting to understand the root causes and impacts of different forms of affect. However, current approaches to affect detection have largely relied upon sensor systems, which are expensive and typically not physically robust to classroom conditions, reducing their potential real-world impact. Work towards sensor-free affect detection has produced detectors that are better than chance, but not substantially better—especially when subject to stringent cross-validation processes. In this paper we present models which can detect student engaged concentration, confusion, frustration, and boredom solely from students' interactions within a Cognitive Tutor for Algebra. These detectors are designed to operate solely on the information available through students' semantic actions within the interface, making these detectors applicable both for driving interventions and for labeling existing log files in the PSLC DataShop, facilitating future discovery with models analyses at scale.

## Keywords

Educational data mining, affective computing, affect detection, boredom, engaged concentration, frustration, confusion, intelligent tutoring system

## 1. INTRODUCTION

In recent years, the log data collected through educational software such as intelligent tutoring systems has been a major resource for the educational data mining community [cf. 12; 33]. In specific, it has been possible to study changes in student learning and engagement over long periods of time by developing models using approaches such as classification or knowledge engineering and applying the models to larger data sets, a process termed "discovery with models." Examples of this research include work to understand which models best predict student learning with an intelligent tutoring system [29; 38], work to find prerequisites within a curriculum [37], work to study the differences in engagement over the course of an entire year between urban, rural, and suburban schools [10], and work to study the differences in disengaged behavior between different tutor lessons [3].

Fewer research studies have focused on affect/academic emotions [30]. It is known that affect interacts with engagement and learning in complex fashions [cf. 7; 11; 21; 22; 26; 35]. However, research of this nature has largely been limited to relatively brief time-windows, on the order of a small number of lab sessions or field sessions. This limitation is due to the methods used in conducting these studies: self-report [cf. 1; 18; 35], retrospective

emote-aloud protocols [cf. 19], field observations [cf. 7; 11; 22], and video observations [19, 21]. Each of these methods has been shown to produce replicable assessments of relevant academic affect, but each method also has limitations in terms of large-scale applicability. Specifically, self-report can disrupt naturally affective processes, and retrospective emote-aloud protocols, like observational methods, are expensive to conduct at large-scale.

A method with the potential to address this limitation is automated detection of affect. Researchers have been investigating affect detection from physiological sensors or vocal patterns for over a decade, and have produced successful detectors for a range of emotions. Such work is reviewed in detail by Calvo and D'Mello [14, 15]. In the domain of educational research, researchers have used sensors to develop detectors for several affective constructs. Litman and Forbes-Riley have found that features of students' voices while engaging in vocal dialogues with tutors can predict students' emotions [27]. D'Mello and Graesser have shown that a combination of body language and facial features, in combination with student interaction with the learning software, can be used to detect learner affect [20]. Muldner, Burleson, and VanLehn have shown that a combination of sensors can be used to detect student delight while learning [28]. Finally, Arroyo and colleagues have shown that sensor-based approaches to affect detection can work in urban schools and classrooms, enabling real-time adaptation to students' affect in an authentic learning setting [1].

However, approaches relying upon sensors are limited in application to data sets for which sensors were present. This limits applicability for schools, where sensor breakage can present a challenge for long-term use. In addition, sensor cost can be an economic challenge for schools, and internet connections may not have sufficient bandwidth to log full physiological sensor data for retrospective analysis.

Hence, to achieve maximum utility of an affect detector for retrospective discovery with models analysis, it is necessary to detect affect without reference to any sensor data. Ideally, such detection will be conducted solely with the type of log file data already being collected at large scale, such as the data being collected in the PSLC DataShop repository [cf. 24].

D'Mello and colleagues presented a first paper on an affect detector developed solely from log files [19]. Modeling student affect in the AutoTutor intelligent tutoring system in a laboratory study, they achieved decent agreement to ground-truth labels provided by human video coders. Their model successfully distinguished frustration from the neutral state approximately 40% better than the base rate (e.g. Kappa = approximately 0.4), and distinguished boredom, confusion, and flow from the neutral state approximately 20% better than chance. However, there were a few limitations in this pioneering study that need to be addressed to make sensor-free detectors of affect maximally useful. First, the detectors' best performance was achieved when distinguishing between specific affective states and the neutral state (e.g. all other affective states were discarded from the data set). The detectors achieved relatively poorer performance (Kappa = 0.163) when attempting to distinguish affective states from each other. Second, they re-sampled the data to eliminate imbalance between classes, and validated their models on the re-sampled data. Re-sampling is an appropriate method for generating unbiased classifiers, but the resultant models should ideally be tested on a non-resampled data set to verify detector effectiveness for future application of the models to data with natural class distribution. Third, their models were cross-validated at the observation level, rather than the student level, providing less information on detector generalizability to new students. Within this paper, we attempt to build on the methods in this pioneering research, while addressing these limitations.

A second paper developing non-sensor-based detectors was presented by Conati and Maclaren, who conducted a laboratory study of affect in the game Prime Climb [18]. In this paper, detectors using a combination of questionnaire and log data were used to predict self-reports of student affect, using a Bayesian framework. The cross-validation in this paper was conducted at the student level, giving information on model applicability to new students. Also, affect was compared using a median-split on binary distinctions (such as the distinction between joy versus distress), avoiding bias that may stem from discarding data that is neither the current affective state being detected nor the neutral state.

As in D'Mello et al. [19], Conati and Maclaren re-sampled the data to eliminate imbalance between classes during training. They validated their models using both the re-sampled distribution and the original distribution [18]. For the re-sampled data, their model was 32% better than the base rate at distinguishing between joy and distress, and 6% better than the base rate at distinguishing between admiration and reproach. However, their models achieved accuracy below the base rate when applied to the original distribution. This result indicates the challenge of achieving appropriate cross-validated performance for unbalanced constructs that are only indirectly reflected in student interaction within learning software.

A third paper developing sensor-free affect detectors was presented by Sabourin, Mott, and Lester, who studied the affect of students using the Crystal Island narrative-centered learning environment, [34]. In this classroom study, as in Conati and Maclaren's laboratory research [18], detectors based on a combination of questionnaire and log data were used to predict self-reports of student affect, using a Bayesian framework. As in [18], cross-validation was conducted at the student level. In addition, all relevant data was considered in model development and evaluation, and models were evaluated using the original data distribution rather than a re-sampled distribution. Their model was 38% better than the base rate at identifying focused students, and 24% better than the base rate at identifying curious students. It was less successful at identifying students who were confused (19% better than base rate), frustrated (14% better than base rate), bored (10% better than base rate), excited (8% better than base rate), although the detectors were better than the base rate for every construct except anxiety (3% worse than base rate). The only limitation for broad applicability of these models is the use of questionnaire measures, which require that a new student be given the same questionnaires for the model to be applied to that student.

A fourth paper, by Lee and colleagues [40], presented a sensor-free detector of confusion in a programming development environment. This detector achieved a very high student-level cross-validated Kappa of 0.86, but it is not clear if this detector was assessing the affective state of confusion or the more general experience of a student having difficulty with the material.

Within this paper, we build automated detectors of affect for Cognitive Tutor Algebra I, a widely used learning environment. In doing so, we restrict ourselves to the data generally available for this learning environment in the PSLC DataShop [cf. 24], making

it feasible to apply the resultant detectors to hundreds of thousands of hours of student data. Ground-truth labels are obtained using field observations of affect [7] conducted using a handheld app for the Android platform, and then synchronized with log files. The detectors are constructed using only log data from student actions within the software occurring at the same time as or before the observations. Affect is known to have different prevalence following specific behaviors [cf. 7; 11; 35], suggesting that a detector that takes this information into account may be more effective than one that does not. By using only information from before and during the observation, our detectors can be used for fail-soft interventions, as well as discovery with models analyses.

# 2. METHODS

## 2.1 Data Collection

Data on student affect was collected from 89 students who were using Cognitive Tutor Algebra I as part of their regular mathematics curriculum. The students were using a lesson on systems of algebraic equations. Cognitive Tutors are a popular type of interactive learning environment now used by around half a million students a year in the USA. In Cognitive Tutors, students solve problems with exercises and feedback chosen based on a model of which skills the student possesses. Cognitive Tutor Algebra has been shown to significantly improve student performance on standardized exams and tests of problem-solving skill [25].

Each of the students studied in this paper were enrolled in one of four classes in a high school in rural Western Pennsylvania. In this school, 67% of students are rated as proficient or higher on the PSSA standardized exam, moderately higher than the state average. Students in this school are 96% Caucasian, typical in rural schools in this region, but higher than the state average. 18% of students are eligible for free or reduced-price lunch, approximately half of the state average. Students studied were approximately balanced in terms of gender.

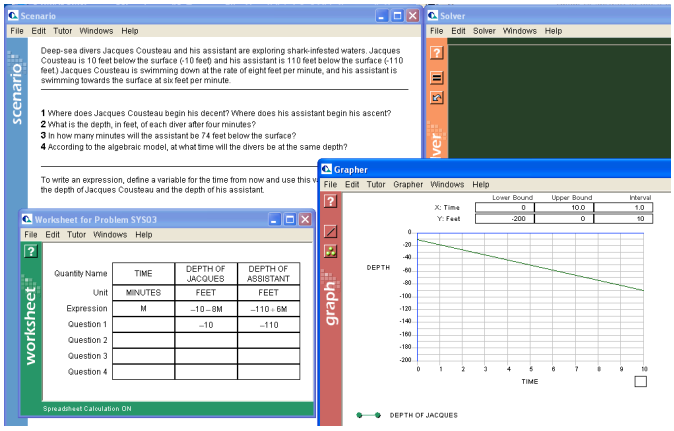



**Figure 1: The Systems of Equations A lesson, from Cognitive Tutor Algebra I, used in this study.**

Two expert field observers coded student affect and engaged/disengaged behaviors as students used the learning software. In this paper, we focus solely on the affect codes, as models of relevant engaged and disengaged behaviors were already available for this tutoring system (see discussion of features below). The coders used software on a Google Android handheld computer, which implemented an observation protocol developed specifically for the process of coding behavior and affect during use of educational software, replicating the protocol in [7]. All coding was conducted by the fourth and fifth authors. These two coders were previously trained in coding behavior and affect by the first author and have achieved inter-rater reliability with the first author of 0.72 (first and fourth authors, affect) and 0.83 (first and fifth authors, behavior [cf. 6]) in previous research conducted with students using other learning environments. This degree of reliability is on par with Kappas reported by past projects that have assessed the reliability of detecting naturally occurring emotional expressions [7; 13; 27; 32].

Observations were conducted in the school's computer laboratory, where students typically use the Cognitive Tutor software. Students were observed across 2 class days. Students were coded in a pre-chosen order, with each observation focusing on a specific student, in order to obtain the most representative indication of student affect possible. At the beginning of each class, an ordering of observation was chosen based on the computer laboratory's layout, and was enforced using the hand-held observation software. Setting up observations took a few minutes at the beginning of each class. A total of 408.51 minutes of observations were conducted across sessions, across the two coders. During this time, 763 observations were conducted across all students, not counting observations of students who were not logged into the software or not present in the classroom, for an average of 8.57 observations per student (SD = 2.84).

Each observation lasted up to twenty seconds, with observation time automatically coded by the handheld observation software. If affect and behavior were determined before twenty seconds elapsed, the coder moved to the next observation. Typically, each student observation involved 5 taps to the handheld screen, with the coder choosing affect and behavior codes from a pair of pop-up menus, and then clicking to confirm their selection. As such, data entry by an experienced coder took approximately 3 seconds per observation.

Each observation was conducted using peripheral vision or side-glances to reduce disruption. That is, the observers stood diagonally behind the student being observed and avoided looking at the student directly [cf. 5; 7; 32], in order to make it less clear when an observation was occurring. This method of observing using peripheral vision was previously found to be successful for assessing student behavior and affect, achieving good inter-rater reliability [cf. 6, 7; 32]. To increase tractability of both coding and eventual analysis, if two distinct affective states were seen during a single observation, only the first state observed was coded. Any affect of a student other than the student currently being observed was not coded.

The observers based their judgment of a student's state or behavior on the student's work context, actions, utterances, facial expressions, body language, and interactions with teachers or fellow students. These are, broadly, the same types of information used in previous methods for coding affect [e.g. 13], and in line with Planalp et al's [31] descriptive research on how humans generally identify affect using multiple cues in concert for maximum accuracy rather than attempting to select individual cues. The judgments of affect were based on a sub-set of the coding scheme used in [7; 21], selected based on importance for learning. Within an observation, each observer coded affect with reference to five categories:

- Boredom
- Confusion
- Engaged concentration (the affect associated with the flow state [cf. 7])
- Frustration
- "?" (which refers to any affect outside the coding scheme, including eureka, delight, and surprise. It also includes cases where it was impossible to code affect, such as when a student went to the bathroom or the software crashed.)

Delight and surprise were removed from the earlier coding scheme in [7; 21], due to the relative rarity of these affective states in prior research [e.g. 7; 21; 32].

Within the observations, the affective states had the following frequencies: boredom was observed 5.9% of the time, engaged concentration was observed 84.5% of the time, frustration was observed 0.9% of the time, and confusion was observed 1.8% of the time. The remaining observations were coded as "?". This distribution of affect is in line with prior studies – engaged concentration is typically the most common affect in classroom learning [cf. 7; 10; 34]. However, confusion, which tends to be relatively rare in most cases, was somewhat less frequent than has been typically seen in previous classroom studies [cf. 7; 10; 34].

## 2.2 Feature Distillation

In order to distill a feature set for detectors of affect, student actions within the software were synchronized to the field observations. Only the types of data available in standard PSLC DataShop log files [cf. 24] were used, towards producing detectors that could be applied retrospectively to existing data at scale.

During data collection, both the handhelds and the educational software server were synchronized to the same internet-time server. Actions during the twenty seconds prior to data entry by the observer were collected as a clip.

A total of 58 features were developed using the student's behavior both during and prior to the 20-second window. Some features were completely about the current action, such as whether it was correct or not. Other features, such as the number of previous actions on the current skill that involved help requests, involved data from the student's past performance. These 58 features were aggregated across the actions within the clip using mean, min, max and sum aggregators, hence a total of 232 features were used in the development of the detectors. Features involving past behavior (such as the number of previous actions on the current skill that involved help requests) are likely to have little change during the course of a clip, but were aggregated in the same fashion for simplicity of implementation.

Using both features on the current clip and features involving past data has the potential to help us detect affect more effectively, as there is evidence that the prevalence of specific affective states is different following specific behaviors [7; 10; 35] during real-world learning.

Features were drawn from two sources:

- Features developed during our group's past work to develop behavior detectors in Cognitive Tutors [cf. 2; 4; 8], averaged across actions in the clip (or min or max across actions), or across actions prior to the clip.
- Prior models of disengaged and engaged behaviors previously developed for this tutor or related tutors [cf. 2; 4; 8; 36]. Engaged and disengaged behaviors are known to precede and co-occur with affect, giving potential leverage for detecting affect.

Examples of features used can be seen in Table 2.

## 2.3 Machine Learning Algorithms

Each affective state was predicted separately – e.g. BORED was distinguished from NOT BORED (e.g. all other affective states), FRUSTRATED was distinguished from NOT FRUSTRATED (e.g. all other affective states), and so on. This resulted in four detectors, one for boredom, confusion, engaged concentration, and frustration respectively.

Each detector was evaluated using six-fold student-level cross-validation [cf. 17; 34]. In this process, students are split randomly into six groups. Then, for each possible combination, a detector is developed using data from five groups of students before being tested on the sixth "held out" group of students. By cross-validating at this level, we increase confidence that detectors will be accurate for new students.

For each construct being detected, a separate student-level cross-validation was conducted, which stratified students based on the dependent variable. This procedure was used in order to guarantee that each fold had a representative number of observations of the majority and minority class. In addition, for unbalanced classes, re-sampling was used to make the class frequency more equal for detector development. However, all goodness calculations were made with reference to the original data set, as in Sabourin et al. [34].

We attempted to fit sensor-free affect detectors using eight common classification algorithms that have been successful for past educational data mining problems, including J48 decision trees, step regression, JRip, Naïve Bayes, and REP-Trees.

Feature selection for machine learning algorithms was conducted using forward selection, where the feature that most improves model goodness is added repeatedly until adding additional features no longer improves model goodness. During feature selection, cross-validated kappa on the original (e.g. non-re-sampled) data set was used as the goodness metric. Prior to feature selection, all features with cross-validated kappa equal to or below zero in a single-feature model were omitted from consideration, as a check on over-fitting.

Detector goodness was assessed using two metrics: Cohen's Kappa [17] and A' [23]. Cohen's Kappa assesses the degree to which the detector is better than chance at identifying which clips involve a specific affective state. A Kappa of 0 indicates that the detector performs at chance, and a Kappa of 1 indicates that the detector performs perfectly. For example, a Kappa of 0.31 would indicate that the detector is 31% better than chance. A' is the probability that the algorithm will correctly identify whether a specific affective state is present or absent in a specific clip. A' is equivalent to both the area under the ROC curve in signal detection theory, and to W, the Wilcoxon statistic [23]. A model with an A' of 0.5 performs at chance, and a model with an A' of 1.0 performs perfectly. In these analyses, A' was computed at the level of clips, rather than students, using the AUC (area under the curve) approximation.

## 3. RESULTS

Detector performance for all four constructs studied was better than chance (see discussion of cross-validation methodology in the previous section), but left room for improvement. Full results are shown in Table 1. For engaged concentration, the best algorithm was K*. The engaged concentration detector achieved an A' of 0.71 and a Kappa of 0.31. For confusion, the best algorithm was JRip. The confusion detector achieved an A' of 0.99 and a Kappa of 0.40. For frustration, the best algorithm was REPTree. The frustration detector achieved an A' of 0.99 and a Kappa of 0.23. For boredom, the best algorithm was Naïve Bayes. The boredom detector achieved an A' of 0.69 and a Kappa of 0.28.

Several of these detectors showed an imbalance between A' and Kappa. Imbalance of this nature typically indicates a detector which is better at getting the relative order between classes correct (in its confidence estimates) than at drawing an optimal line between classes. Using detectors of this nature, whether for intervention or discovery with models analyses, will be more effective if confidence is taken into account.

Features automatically selected for each of the detectors during machine learning are listed in Table 2. Full detail on models, including runnable versions of the models (for RapidMiner 4.6) can be found in the PSLC DataShop [24], in data set "Baker – Closing the Loop on Gaming – Hopewell Spring 2011", at (https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=479). This data set also includes all data used in the analysis, distilled features used to develop the detectors, data from the field observations synchronized with the student interaction data, and the detector's predictions for actions not initially labeled through the field observations.

The features for engaged concentration involve actions where the student was more likely to have a history of few errors and help requests on the skills in the clip. When a student is in engaged concentration and requests help, they typically do not follow it up with an error (e.g. they read the hints carefully; while they might also have been gaming the tutor's hints, gaming typically results in some proportion of errors, as students do not read hints carefully). These features suggest a student who is closely engaged and working effectively.

| | A' | Kappa |
|---|---|---|
| Engaged Concentration | 0.71 | 0.31 |
| Confusion | 0.99 | 0.40 |
| Frustration | 0.99 | 0.23 |
| Boredom | 0.69 | 0.28 |
| Average Across Constructs | 0.85 | 0.30 |

**Table 1: The goodness of each final model, under student-level cross-validation, for the original data set.**

| **Engaged Concentration** |
|---|
| The minimum number of previous incorrect actions and help requests for any skill in the clip. |
| Among the skills involved in the clip, the minimum value for previous incorrect actions and help requests for that skill. |
| The duration (in seconds) of the fastest action in the clip. |
| The percentage of clip actions involving a hint followed by an error. |

| **Boredom** |
|---|
| The average time the student took to respond on the current step prior to the clip, averaged across all the actions with a clip. |
| The average time the student took to respond, unitized across time taken by all students on the same problem steps, within sequences of three actions in a row. |
| The maximum product of the probability of moment-by-moment learning P(J) [9], and the probability of guess P(G) calculated using the contextual guess model [4] for any action in the clip. This can be interpreted as actions where the student learned after guessing. |
| The maximum number of previous incorrect actions and help. requests for any skill in the clip. |

| **Confusion** |
|---|
| The percentage of clip actions involving actions taking longer than 5 seconds after two incorrect answers. |
| The percentage of actions in the clip that were hint requests. |
| The minimum number of previous incorrect actions for any skill in the clip. |
| The maximum product of the probability of guess P(G) as computed using contextual guess model [4], across sequences of three actions in a row. |
| The average time the student took to respond, unitized across time taken by all students on the same problem steps, within sequences of five actions in a row that were correct. |

| **Frustration** |
|---|
| The percent of past actions on the skills involved in the clip that were incorrect. |
| Were there any actions in the clip where the student made a wrong answer rather than requesting help when their probability of knowing the skill was under 0.7? |

**Table 2. The features in the final detectors of each construct.**

The features for confusion suggest a struggling student [cf. 26]. A confused student is more likely to have a pattern of slower actions after making two errors and tend to have a history of more incorrect actions and help requests. Furthermore, his or her correct actions are relatively more likely to represent guesses, using the contextual guess model from [4]. On the other hand a student who is not confused tends to be able to successfully answer 5 items in rows, working slowly.

The features for frustration involve incorrect actions and help avoidance. In particular, frustrated students tend to have a history of past incorrect actions and help requests. Curiously, frustrated students are more likely to avoid help and make errors when they do not know the skill. It is unclear whether this behavior is a result of frustration, or whether it is perhaps a cause of frustration.

The features for boredom are interestingly different than the features for other constructs. Bored students were more likely to guess than other students. Interestingly, though, they were also relatively likely to learn from their guesses. Compared to other students, bored students were relatively less likely to have a history of many errors and help requests. In addition, students who were bored had a past history of working slowly, and worked slowly while they were bored, across multiple actions within the tutor software.

# 4. DISCUSSION AND CONCLUSIONS

In this paper, we have presented automated detectors that are a step towards identifying student affect solely from log files, in a Cognitive Tutor for Algebra. These detectors are better than chance at identifying engaged concentration, confusion, frustration, and boredom, among a population of students using the Cognitive Tutor as part of their regular mathematics classes.

These detectors achieve goodness values that are moderately better than past values obtained through sensor-free detectors, when averaged across constructs. In this study, the average detector Kappa was 0.30. The detectors closest in validation to this study within D'Mello et al. [19], albeit in a different domain, achieved an average Kappa of 0.16. The detectors closest in validation to this study within Conati & Maclaren [18] achieved an average accuracy below the base rate, and detectors validated on re-sampled data achieved an average accuracy that was 19% better than the base rate (approximately comparable to Kappa of 0.19). The detectors in Sabourin et al. [34], validated in the same fashion as these detectors, achieved an average accuracy that was 16% better than the base rate (approximately comparable to Kappa of 0.16). Individual detectors from previous studies performed better than the detectors presented here (e.g. frustration in [19], focused/engaged concentration in [34]), but on the average the detectors presented here performed better than detectors presented in previous papers. While comparison of model goodness obtained in different software platforms, age groups, and populations should be done with caution, the detectors presented here appear to represent further progress towards effective, sensor-free detectors of affect.

It is possible that at least part of this progress is the result of a greater degree of feature engineering in this detector's development, including the use of features previously used to detect disengaged behaviors, and existing models of several potentially relevant constructs such as guessing [4]. These results suggest that by using both the detectors of disengaged behaviors known to be associated with affect as features and the features used to produce those detectors, increased detector goodness can be obtained with acceptable construct validity.

At the same time, our affect detectors are clearly still imperfect. These new features have only achieved 30% of potential progress towards perfect detection, and, while perfect detection is probably infeasible (after all, even expert coders only achieve Kappa values around 0.6 or 0.7), there is clearly substantial room for improvement. Further work should consider further feature engineering, and potentially alternate methods for aggregating data. Continued improvement in terms of feature engineering may be supported by further research on the behaviors that correspond to specific affective states [cf. 7; 11; 35].

In addition, there is considerable work needed in the area of cross-validation. The detectors presented here are developed and validated for a single, fairly homogenous population. As such, their validity for the broad and diverse population of learners using Cognitive Tutor Algebra in the USA has not yet been fully established. Likewise, the detectors are developed within the context of a single Cognitive Tutor lesson. As such, the detector's validity for new curricular materials has not been established. The detectors may indeed be generalizable and usable in new contexts, as past detectors of disengaged behaviors have often been found to be (for instance, in their use within the detectors presented within this paper), but establishing generalizability will be an important area of future work.

One positive note for the applicability of these detectors in other populations and domains is that the behaviors identified by each detector have reasonable construct validity, suggesting that the detectors may be less accurate in these contexts, but are unlikely to provide meaningless predictions. For this reason, it may still be appropriate to use these detectors in discovery with models analyses, with the expectation that the strength of correlations may be reduced, but that findings with high strength are unlikely to be wholly spurious. The detectors can also be used immediately in the development of detectors of other constructs, as behavior detectors were used here. In these cases, the validity of the detectors is shown by their relevance to detecting other constructs. Thus, though the detectors are imperfect, they still may prove a useful component for EDM research. As these detectors predict affect solely using log file data, they can be applied to existing data from Cognitive Tutor Algebra in the PSLC DataShop and elsewhere. As hundreds of thousands of students use this software each year, we believe that many analyses can be accomplished with these detectors and look forward to working with colleagues to accomplish this goal.

Similarly, it may be possible to incorporate these detectors into the Cognitive Tutor software for fail-soft interventions, which could be used to advance learning outcomes.

In the long-term, detectors of this nature are likely to provide a useful tool for understanding and automatically adapting to differences in learner affect. We see the work here as an incremental step, following on the pioneering work of D'Mello and colleagues [19], Conati and Maclaren [18], and Sabourin, Mott, and Lester [34] towards this goal.

# 5. ACKNOWLEDGMENTS

This research was supported by grant "Toward a Decade of PSLC Research: Investigating Instructional, Social, and Learner Factors in Robust Learning through Data-Driven Analysis and Modeling," National Science Foundation award #SBE-0836012, and by grant #OPP1048577 from the Bill & Melinda Gates Foundation. We also thank Leigh Ann Sudol, Ma. Mercedes T. Rodrigo, and Sidney D'Mello for helpful comments and suggestions.

# 6. REFERENCES

[1] Arroyo, I., Woolf, B.P., Cooper, D., Burleson, W., Muldner, K., and Christopherson, R. Emotion Sensors Go To School. In V. Dimitrova, R. Mizoguchi, B. Du Boulay, and A. Graesser (eds.) *14th International Conference on Artificial Intelligence In Education*, 2009.

[2] Baker, R.S.J.d. Modeling and Understanding Students' Off-Task Behavior in Intelligent Tutoring Systems. *Proceedings of ACM CHI 2007: Computer-Human Interaction*, pp. 1059-1068, 2007.

[3] Baker, R.S.J.d. Differences Between Intelligent Tutor Lessons, and the Choice to Go Off-Task. *Proceedings of the 2nd International Conference on Educational Data Mining*, pp. 11-20, 2009.

[4] Baker, R.S.J.d., Corbett, A.T., and Aleven, V. More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, pp. 406-415, 2008.

[5] Baker, R.S.J.d., Corbett, A.T., Koedinger, K.R., and Wagner, A.Z. Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System". *Proceedings of ACM CHI 2004: Computer-Human Interaction*, pp. 383-390, 2004.

[6] Baker, R.S.J.d., Corbett, A.T., and Wagner, A.Z. Human Classification of Low-Fidelity Replays of Student Actions. *Proceedings of the Educational Data Mining Workshop at the 8th International Conference on Intelligent Tutoring Systems*, pp. 29-36, 2006.

[7] Baker, R.S.J.d., D'Mello, S.K., Rodrigo, M.M.T., and Graesser, A.C. Better to Be Frustrated than Bored: The Incidence, Persistence, and Impact of Learners' Cognitive-Affective States during Interactions with Three Different Computer-Based Learning Environments. *International Journal of Human-Computer Studies*, 68 (4), pp. 223-241, 2010.

[8] Baker, R.S.J.d., and de Carvalho, A.M.J.A. Labeling Student Behavior Faster and More Precisely with Text Replays. *Proceedings of the 1st International Conference on Educational Data Mining*, pp. 38-47, 2008.

[9] Baker, R.S.J.d., Goldstein, A.B., Heffernan, N.T. Detecting Learning Moment-by-Moment. To appear *in International Journal of Artificial Intelligence in Education*, in press.

[10] Baker, R.S.J.d., and Gowda, S.M. An Analysis of the Differences in the Frequency of Students' Disengagement in Urban, Rural, and Suburban High Schools. *Proceedings of the 3rd International Conference on Educational Data Mining*, pp. 11-20, 2010.

[11] Baker, R.S.J.d., Moore, G., Wagner, A., Kalka, J., Karabinos, M., Ashe, C., and Yaron, D. The Dynamics Between Student Affect and Behavior Occurring Outside of Educational Software. *Proceedings of the 4th bi-annual International Conference on Affective Computing and Intelligent Interaction*, pp. 14-24, 2011.

[12] Baker, R.S.J.d., and Yacef, K. The State of Educational Data Mining in 2009: A Review and Future Visions. *Journal of Educational Data Mining*, 1 (1), pp. 3-17, 2009.

[13] Bartel, C.A., and Saavedra, R. The collective construction of work group moods. *Administrative Science Quarterly*, 45, pp. 197-231, 2001.

[14] Calvo, R.A., and D'Mello, S.K. Affect Detection: An Interdisciplinary Review of Models, Methods, and their Applications. *IEEE Transactions on Affective Computing*, 1 (1), pp. 18-37, 2010.

[15] Calvo, R.A., and D'Mello, S.K. eds. New Perspectives on Affect and learning technologies. New York: Springer. 2011.

[16] Cleary, J.G., and Trigg, L.E. K*: An Instance- based Learner Using an Entropic Distance Measure. *Proceedings of the 12th International Conference on Machine learning*, pp. 108-114, 1995.

[17] Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20 (1), pp. 37-46, 1960.

[18] Conati, C., and Maclaren, H. Empirically building and evaluating a probabilistic model of user affect. *User Modeling and User-Adapted Interaction*, 19, pp. 267-303, 2009.

[19] D'Mello, S.K., Craig, S.D., Witherspoon, A. W., McDaniel, B. T., and Graesser, A. C. Automatic Detection of Learner's Affect from Conversational Cues. *User Modeling and User-Adapted Interaction*, 18 (1-2), pp. 45-80, 2008.

[20] D'Mello, S.K., and Graesser, A.C. Multimodal semi-automated affect detection from conversational cues, gross body language, and facial features. *User Modeling and User-adapted Interaction*. 20 (2), pp. 147-187, 2010.

[21] D'Mello, S.K., Taylor, R., and Graesser, A.C. Monitoring Affective Trajectories during Complex Learning. In D.S. McNamara & J.G. Trafton (eds.) *Proceedings of the 29th Annual Cognitive Science Society*, pp. 203-208, 2007.

[22] Dragon, T., Arroyo, I., Woolf, B.P., Burleson, W., El Kaliouby, R., and Eydgahi, H. Viewing Student Affect and Learning through Classroom Observation and Physical Sensors. *International Conference on Intelligent Tutoring Systems 2008*, pp. 29-39, 2008.

[23] Hanley, J., and McNeil, B. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143, pp. 29-36, 1982.

[24] Koedinger, K. R., Baker, R., Cunningham, K., Skogsholm, A., Leber, B., and Stamper, J. A data repository for the EDM community: The PSLC DataShop. In C. Romero, S. Ventura, M. Pechenizkiy, and R.S.J.d. Baker (eds.) *Handbook of Educational Data Mining*. pp. 43-55, Boca Raton, FL: CRC Press, 2010.

[25] Koedinger, K.R., and Corbett, A.T. Cognitive Tutors: Technology bringing learning science to the classroom. In K. Sawyer (ed.) *The Cambridge Handbook of the Learning Sciences*, pp. 61-78, Cambridge, UK: Cambridge University Press, 2006.

[26] Lee, D.M., Rodrigo, M.M., Baker, R.S.J.d., Sugay, J., and Coronel, A. Exploring the Relationship Between Novice Programmer Confusion and Achievement. *Proceedings of the 4th bi-annual International Conference on Affective Computing and Intelligent Interaction*, 2011.

[27] Litman, D.J., and Forbes-Riley, K. Recognizing Student Emotions and Attitudes on the Basis of Utterances in Spoken Tutoring Dialogues with both Human and Computer Tutors. *Speech Communication*, 48 (5), pp. 559-590, 2006.

[28] Muldner, K., Burleson, B., and VanLehn, K. "Yes!": Using tutor and sensor data to predict moments of delight during instructional activities. *Proceedings of the International Conference on User Modeling and Adaptive Presentation (UMAP'10)*, pp. 159-170, 2010.

[29] Pardos, Z.A., and Heffernan, N. T. Using HMMs and bagged decision trees to leverage rich features of user and skill from

an intelligent tutoring system dataset. To appear in the *Journal of Machine Learning Research W & CP*, in press.

[30] Pekrun, R., Goetz, T., Titz, W., and Perry, R. P. Academic emotions in students' self-regulated learning and achievement: A program of quantitative and qualitative research. *Educational Psychologist*, 37, pp. 91-106, 2002.

[31] Planalp, S., DeFrancisco, V. L., and Rutherford, D. Varieties of cues to emotion in naturally occurring situations. *Cognition and Emotion*, 10 (2), pp. 137-153, 1996.

[32] Rodrigo, M.M.T., Baker, R.S.J.d., D'Mello, S., Gonzalez, M.C.T., Lagud, M.C.V., Lim, S.A.L., Macapanpan, A.F., Pascua, S.A.M.S., Santillano, J.Q., Sugay, J.O., Tep, S., and Viehland, N.J.B. Comparing Learners' Affect While Using an Intelligent Tutoring Systems and a Simulation Problem Solving Game. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, pp. 40-49, 2008.

[33] Romero, C., and Ventura, S. Educational Data Mining: A Review of the State-of-the-Art. *IEEE Transaction on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 6, pp. 601-618, 2010.

[34] Sabourin, J., Mott, B., and Lester, J. Modeling Learner Affect with Theoretically Grounded Dynamic Bayesian Networks. *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction*, pp. 286-295, 2011.

[35] Sabourin, J., Rowe, J., Mott, B., and Lester, J. When Off-Task in On-Task: The Affective Role of Off-Task Behavior in Narrative-Centered Learning Environments. *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, pp. 534-536, 2011.

[36] Shih, B., Koedinger, K.R., and Scheines, R. A response time model for bottom-out hints as worked examples. *Proceedings of the 1st International Conference on Educational Data Mining*, pp. 117-126, 2008.

[37] Vuong, A., Nixon, T., and Towle, B. A Method for Finding Prerequisites Within a Curriculum. *Journal of Educational Data Mining*, pp. 211-216, 2011.

[38] Yu, H.-F., Lo, H.-Y., Hsieh, H.-P., Lou, et al. Feature Engineering and Classifier Ensemble for KDD Cup 2010. *Proceedings of the KDD Cup 2010 Workshop*, pp. 1–16, 2010.

# Using Edit Distance to Analyse Errors in a Natural Language to Logic Translation Corpus

Dave Barker-Plummer
Center for the Study of Language and Information
Stanford University
Stanford, CA 94305, USA
dbp@stanford.edu

Robert Dale
Center for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
Robert.Dale@mq.edu.au

Richard Cox
Faculty of Information Technology
Monash University
Clayton, VIC 3168, Australia
rcox@inf.ed.ac.uk

## ABSTRACT

We have assembled a large corpus of student submissions to an automatic grading system, where the subject matter involves the translation of natural language sentences into propositional logic. Of the 2.3 million translation instances in the corpus, 286,000 (approximately 12%) are categorized as being in error. We want to understand the nature of the errors that students make, so that we can develop tools and supporting infrastructure that help students with the problems that these errors represent.

With this aim in mind, this paper describes an analysis of a significant proportion of the data, using edit distance between incorrect answers and their corresponding correct solutions, and the associated edit sequences, as a means of organising the data and detecting categories of errors. We demonstrate that a large proportion of errors can be accounted for by means of a small number of relatively simple error types, and that the method draws attention to interesting phenomena in the data set.

## 1. INTRODUCTION

As reported in [2], we have developed a large corpus of student submissions to an automatic grading system, where the subject matter involves the translation of natural language (NL) sentences into first-order logic (FOL). The translation exercises are a subset of the exercises in *Language, Proof and Logic* (LPL; [6]), a courseware package consisting of a textbook together with desktop applications which students use to complete exercises.[1] The translation exercises contain a total of 275 distinct translatable sentences; an abridged example of such an exercise is shown in Figure 1.[2] We refer to each translation of a sentence submitted by a student as a **translation instance**; the corpus contains 4.5 million translation instances, collected from more than 55,000 individual students over the period 2001–2010. In this paper, we focus on that subset of the data concerned with translations into propositional logic; this accounts for 2,340,306 translation instances. The Grade Grinder, our automatic grading system, categorizes 286,106 of these (approximately 12%) of these instances as being in error; Figure 2 shows some common incorrect solutions for the first sentence in the exercise shown in Figure 1.

Currently, our grading system simply indicates whether or not the student has obtained a correct answer. Our interest is in (a) developing tools and supporting infrastructure that can help students with these errors by providing better feedback; and (b) using the evidence from the errors that students make to drive broader pedagogical improvements. In order to achieve these goals, we first need to acquire an understanding of the kinds of errors that students make.

Of course, any teacher of logic has intuitions about what aspects of logic students find hard and why. Any good logic textbook embodies a distilling of what is essentially folk wisdom based on teacher experience. In contrast, our goal is to use our large dataset of errors as part of an evidence-based approach to diagnosing, assisting and correcting a wide range of errors. There have been data-driven studies in the past (see, for example, our own [3]), but these have been based on relatively small amounts of data, and so there is a risk that they are essentially anecdotal.

Our goal, then, is to see what we can learn by looking at larger and more comprehensive sets of student behaviour in this domain. In this paper, we look at how **edit distance** can be used as a tool for characterising the nature of the errors in our dataset. We demonstrate that a large proportion of errors can be accounted for by means of relatively simple error types, and discuss how some of these errors correspond to specific kinds of problems that students struggle with.

The structure of the remainder of this paper is as follows. In Section 2, we briefly review related work, describe the data we work with, and outline our general approach to its analysis. In Section 3, we provide some summary statistics on the subset of the data that it is the focus of the present paper. We then move on in Sections 4 and 5 to a detailed analysis of the errors in this data set, and make some observations based on our analyses. Finally, in Section 6, we draw some conclusions and outline our future plans.

## 2. BACKGROUND

### 2.1 Related Work

Students have particular difficulty with the formalisation tasks that underpin logic, and so it is not surprising that there is a body of work that seeks to understand the nature of these difficulties, often with the aim of improving on-line tutoring systems; for example, Barnes et al. [5] studied students' logic learning trajectories in order to extend Deep

[1] See `http://lpl.stanford.edu`.
[2] The '$\Delta$' column is explained later.

➹ **Exercise 7.12** (Translation) Translate the following English sentences into FOL. Your translations will use all of the propositional connectives.

(1) *If* a *is a tetrahedron then it is in front of* d.
(2) a *is to the left of or right of* d *only if it's a cube.*
(3) c *is between either* a *and* e *or* a *and* d.
...
(19) a *is large just in case* d *is small.*
(20) a *is large just in case* e *is.*

Figure 1: An example exercise (7.12) from LPL

| Ex. | N | Correct | Incorrect | Δ |
|---|---|---|---|---|
| 1 | 429 | $\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, d)$ | $\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, b)$ | 1 |
| 2 | 254 | $\neg\mathsf{FrontOf}(a, d) \rightarrow \neg\mathsf{Tet}(a)$ | $\mathsf{FrontOf}(a, d) \rightarrow \mathsf{Tet}(a)$ | 2 |
| 3 | 160 | $\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, d)$ | $\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(d)$ | 1 |
| 4 | 96 | $\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, d)$ | $\mathsf{Tet}(a) \leftrightarrow \mathsf{FrontOf}(a, d)$ | 1 |

Figure 2: Correct and incorrect solutions to Exercise 7.12.1

Thought, a logic proof tutor, by adding a hint generator.

The work carried out by various groups has helped to identify specific categories of error that recur in formalisation tasks.

**Logical misunderstandings** are cognitive misconceptions, such as when a student substitutes the biconditional (*if and only if*) for material implication (*if . . . then*), or when they reverse the antecedent and consequent of an implication, writing $A \rightarrow B$ when the correct answer is $B \rightarrow A$. These misunderstandings are often related to what we might think of as **natural language transfer errors**, where some property true of the natural language sentence, typically to do with its structure or ordering of elements, is also assumed to be true of the formal translation. An early study by Clement, Lochhead and Monk [8] investigated such translation difficulties in mathematics: students were instructed to write an equation representing the statement *There are six times as many students as professors at this university*, and were told to use $S$ for the number of students and $P$ for the number of professors. The most common error consisted of reversing the variables in the equation, writing 6S = P. An explanation of this phenomenon offered by Clement et al. was **word-order matching**, in which the student orders terms in their equation in a way that matches the order of keywords in the problem statement.

A more general source of difficulty for students lies in the stance they must adopt towards formal language, which, compared to everyday discourse, is more concise, less redundant, and has high information density and precision. Otherwise everyday terms are often used in formal contexts with quite specific meanings; Pimm [13] provides numerous examples of such 'borrowed terms' in mathematics (e.g., *face*, *mean*, *real*, *natural*). Consequently, a student may bring expectations in regard to the use of particular terms that are not in fact borne out.

Of course, errors which are not unique to formalisation also manifest themselves in formalisation tasks. **Slips** [14] are such a type of error; in the context of the translation of natural language into logic, an example would be an attentional lapse resulting in a student using the wrong letter to denote a constant (e.g., b instead of d).

Our own work to-date has focussed upon characterizing logic translation tasks in terms of the grammatical and other features of the English sentences to be translated, and upon the elucidation of the nature of errors made by students on those tasks. In previous work we have carried out a detailed analysed of the errors that students make when performing translation tasks using data from just one of the LPL exercises [3], and we have compared the errors that students make when translating into logic as compared to diagrammatic form [9], together with the effects of visual and spatial content [4]. We have also looked at students' individual learning trajectories by examining their repeated submissions to the Grade Grinder [1].

## 2.2 The Data

LPL contains 33 exercises that involve translations of natural language into logic. Each exercise involves multiple sentences to be translated, so these 33 exercises consist of 275 distinct natural language sentences that students are asked to translate. A complete list of the sentences to be translated, together with the number of translation instances we have for each natural language sentence, and the proportion of instances that were deemed by the Grade Grinder to be in error, is provided in [2], along with other detailed statistics.

When a student submits an exercise, the Grade Grinder assigns a **status** to each sentence in that exercise. An exercise is considered correct if every sentence in the submission is assigned a status of **correct**, which indicates that the sentence is provably equivalent to a gold-standard correct answer. Other common statuses include **incorrect**, which indicates that the sentence is not provably correct, and **ill-formed**, which indicates that the student's submission is not a well-formed formula of first-order logic.

Each entry in our corpus records the identity of the sentence being translated, a unique identifier for the student, the particular string submitted by the student, and the status that it was assigned by the Grade Grinder. Auxilliary information concerning the time at which the submission was made, and other similar meta-information, is also recorded, but plays no role in the current study.

LPL introduces the language of first-order logic in two stages. In the first, the quantifier-free fragment (propositional logic) is introduced, while consideration of the full first-order language is deferred until the second stage. All of the data for this study is drawn from exercises in propositional logic, i.e., from Chapters 1–8 of the textbook; this results in a subcorpus of 286,106 translation instances.

## 2.3 Approach

In our earlier explorations of the data [1; 3; 4; 9], we used a somewhat labor-intensive approach to uncovering the nature of the errors that students make: this involved writing exercise-specific regular expressions to pick out relevant properties of each submitted answer. Such an approach is appropriate for initial forays into large datasets, since it gives a sense of the range and variety of the data; but it doesn't scale up to data collections like the one focussed on

here. We need a way to manage the dataset that provides some structure, and provides some means for determining how subsequent more-detailed analyses might be prioritised.

Given our pedagogical aims, a sensible strategy is to focus on what is common and repeated in the data; if there are some mistakes that many students make, it obviously makes sense to develop techniques to address these issues, rather than focussing on much rarer errors whose probability of reoccurrence is small. But this immediately presents us with a problem. The natural language sentences that students are asked to translate are deliberately and necessarily quite varied, often combining a variety of phenomena both in their surface forms and in their corresponding logical translations. This is entirely appropriate if we want to test a student's ability to deal with non-trivial formalisation problems that are not unidimensional; but this property of the data also makes it difficult to identify and characterise the important features that are shared across instances of error. Put simply: any given incorrect solution provided by a student may simultaneously represent a number of different errors, and the same error or misunderstanding may manifest itself in different sentences in different ways.

Our approach is to seek a method of characterising, for any pairing of a correct answer and an incorrect translation provided by a student, the delta between the two. This delta captures the essence of the particular error or set of errors that the student made; we can then look for similarities across the deltas found in different contexts, with the aim of identifying common problems that recur. Essential to making this work is the availability of a representation that abstracts out just the right characteristics of the delta.

An approach to characterising error that has been used in other domains—most notably in the context of natural language spelling correction—is that of **edit distance** [12]. The idea here is much-studied and well-known: the difference between two objects can be characterised by a sequence of edit operations (typically, the insertion, deletion and transposition of elements), and the length of this sequence is the edit distance between the two objects.[3] In the context of spelling correction, the objects are strings whose constituent elements are characters; the edit operations are therefore the insertion, deletion and transposition of characters. The misspelling *teh* is then correctable to *the* by a single transposition, giving an edit distance of 1. An oft-cited finding is that 80% of spelling errors are edit distance 1 from their corresponding correct forms [10], supporting a view that most errors are simple; in the interface to a spelling correction program, this provides a basis for ordering potential corrections in terms of increasing edit distance.

Technically, the approach just described uses the notion of **string edit distance**. However, our sense is that the tree structure of well-formed logic sentences will play a role in characterising the nature of the errors that students make; we already saw hints of this in our earlier work [3], where we identified antecedent–consequent reversal (where, effectively, two subtrees in a formula are transposed) as a common error type. Consequently, rather than string edit distance, we use **tree edit distance**. Here, the edit distance is defined between ordered labelled trees, i.e., trees

[3]Substitution is also sometimes considered an atomic operation, but this can be achieved by a deletion followed by an insertion.

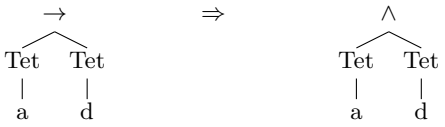



Figure 3: Relabelling

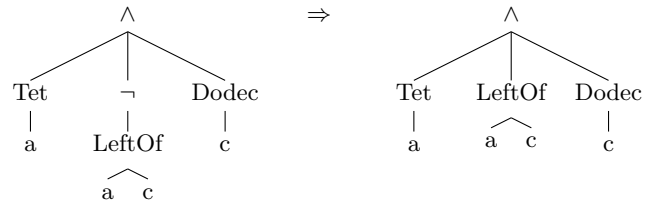



Figure 4: Node deletion/insertion

in which each node has an associated label, and the order of the children of a node is considered significant. We will use the term **tree** as a shorthand for ordered, labelled trees. The edit distance between two trees is defined as the minimum number of edit operations that must be applied to one tree, the **source**, to make it identical to another, the **target**. Critical to this definition is a set of edit operations that we have available to make this transformation. In our formulation, which is typical, three edit operations are available: **relabeling**, in which the label on the node of a tree is changed; **deletion**, in which a node is deleted from the tree, with any children of the deleted node becoming children of the deleted node's parent (in order); and **insertion**, the inverse of deletion. Figure 3 shows a relabelling operation applied to a tree; Figure 4 shows a case where a node has been deleted in the move from left-to-right, or equivalently inserted from right-to-left.

The edit distance problem in trees has been studied extensively (see [7] for a survey); our implementation is of the algorithm due to Klein [11]. More efficient algorithms exist, but the additional complexity of these algorithms is not justified in our context primarily because our trees are quite small.

It is important to note that translation exercises do not have unique solutions. Student submissions are graded as correct if they are provably equivalent to a gold-standard correct answer known to the system. Consequently, the corpus contains a set of submitted correct answers in addition to the set of submitted incorrect answers. This presents a problem for our approach: which of the possible correct answers should we compute the distance from when considering a given incorrect answer? Our response to this problem is to choose the correct answer for which the edit distance is minimized. This strategy essentially embodies the assumption that simple errors are more likely.

Figure 2 shows some example pairs for translations of Sentence 7.12.1 in Figure 1. Example 1 demonstrates an error with edit distance 1, as indicated by the column headed '$\Delta$'. Here, the student has used the constant `b` when `a` was required by the correct answer. Note that a different correct answer is used in Example 2: a simple description of this error is that the student has reversed the order of the antecedent and consequent, but an alternative explanation

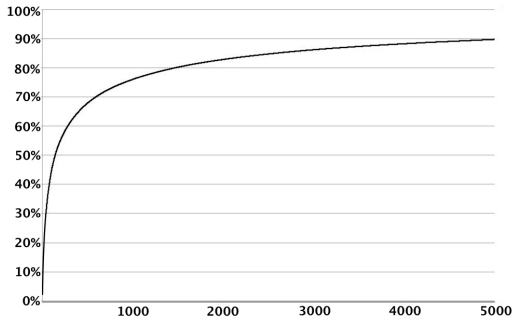



Figure 5: Cumulative coverage by error type

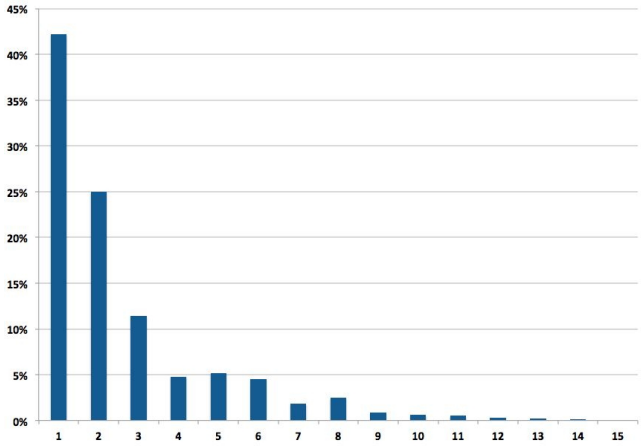



Figure 6: Proportions of data by edit distance

is the one shown here, which is that they have omitted negation symbols from the contrapositive of the answer in Example 1. This transformation has an edit distance of 2, and is 'cheaper' than the corresponding reversal.

## 3. ERROR DISTRIBUTION

An **error instance** is an occasion of a particular student submitting a particular incorrect translation for a given NL sentence; an **error type** is a pairing of an incorrect translation with the correct translation of that sentence, abstracted away from its particular occurrences in the data. The point here is that the data manifests varying numbers of instances for each distinct error type: some incorrect translations are very common (i.e., many students provide the same incorrect translation) and some are rare (i.e., only one or two students produce that particular error).

Our 286,106 error instances are distributed across a total of 27,151 error types, but the distribution is very skewed: the 19 most frequent error types account for 20% of the data, and 142 types account for 50% of the data. Figure 5 shows the cumulative coverage of the error types in order of their frequency in the data.

A note is in order here in regard to our use of absolute counts of errors, rather than proportions of incorrect solutions for each translation. It might be thought that the latter would provide a more realistic measure of where the difficulties lie for students: if only 10 students attempt one question and nine get it wrong, this suggests that the question is more difficult than one where 1000 students attempt the question and 500 get it wrong, despite the larger absolute count of errors in the second case. However, the fact remains that some sentences are translated more often than others; and if we want to maximise the impact of our work, it is the questions which generate high numbers of errors, irrespective of the number of attempts, that are most important. Consequently, in this paper we report statistics in terms of absolute counts, although for other purposes proportions will be more informative.

We note that the most frequent error types in the data are edit distance 1 from their respective correct translations. We call such errors **unit errors** or **units**. Unit errors account for 42.24% of the incorrect answer instances, and errors with edit distance 2 account for a further 24.97% of the incorrect answer instances, for a total of 67.22%. Figure 6 breaks down the error instances by edit distance. There is a long tail, and so the graph in this figure is truncated: the largest edit distance represented in the corpus is 73 (with one instance).

Given the preponderance of relatively simple errors, in the remainder of this paper we focus on the simple and easily detectable edit sequences as a means of organising and analysing the data. This is not to say that the more complex errors are not important; they still account for a significant proportion of the data. However, the analyses required to identify the relevant patterns are correspondingly more complex. Our strategy is therefore to explore the simpler errors first, with the possibility of subsequently factoring these elements out from the more complex errors to further assist in managing the complexity.

## 4. HOMOGENEOUS EDITS

Our goal is to use edit distance as a way of organising the errored data in the corpus. We are interested in two things here, both of which are explored further below.

1. We want to see whether edit distance as a metric provides a useful way to organise and analyse the data; for example, we might expect that it would allow us to organise the data in terms of the complexity of the errors contained.

2. In terms of exploring the particular errors that students make, it is conceivable that the edit sequences required to transform an error into its corrected form might reveal something about the nature of the errors made.

As a first step, we organise relabellings into three subcategories: $\text{RELABEL}_{Conn}$, $\text{RELABEL}_{Const}$, and $\text{RELABEL}_{Pred}$. The basis for this categorization is the syntactic category of the tokens that participate in the relabeling. The tree edit distance algorithm is domain-independent, and thus it has no information about the roles of the symbols at the nodes of the trees. At the logical level, we distinguish between **constant symbols**, which serve to name objects, **predicate symbols**, which serve to name properties and relations, and **logical connectives**, which connect together sentences into larger sentences. When a node relabeling operation involves symbols which are both of the same syntactic category, then

| Ex. | Correct | Incorrect | Instances | Edit Sequence |
|---|---|---|---|---|
| 1 | $\neg\mathsf{Between}(c,a,b) \wedge \neg(\mathsf{FrontOf}(c,a) \vee \mathsf{FrontOf}(c,b))$ | $\neg\mathsf{Between}(c,a,b) \vee \neg(\mathsf{FrontOf}(c,a) \wedge \mathsf{FrontOf}(c,b))$ | 521 | [RELABEL$_{Conn}$ $\vee$ at $\langle\rangle$ to $\wedge$ at $\langle\rangle$, RELABEL$_{Conn}$ $\wedge$ at $\langle.2.1\rangle$ to $\vee$ at $\langle.2.1\rangle$] |
| 2 | $(\mathsf{LeftOf}(a,d) \vee \mathsf{RightOf}(a,d)) \rightarrow \mathsf{Cube}(a)$ | $(\mathsf{LeftOf}(a,d) \vee \mathsf{RightOf}(a,d)) \rightarrow \mathsf{Cube}(d)$ | 899 | [RELABEL$_{Const}$ d at $\langle.2.1\rangle$ to a at $\langle.2.1\rangle$] |
| 3 | $\mathsf{SameCol}(b,e) \wedge \mathsf{SameRow}(e,d) \wedge \mathsf{SameCol}(d,a)$ | $\mathsf{SameCol}(b,e) \wedge \mathsf{SameCol}(e,d) \wedge \mathsf{SameCol}(d,a)$ | 769 | [RELABEL$_{Pred}$ SameCol at $\langle.2\rangle$ to SameRow at $\langle.2\rangle$] |
| 4 | $\mathsf{Cube}(a)$ | $\mathsf{Cub}(a)$ | 1 | [RELABEL$_{Pred}$ Cub at $\langle\rangle$ to Cube at $\langle\rangle$] |
| 5 | $\neg\mathsf{Cube}(b) \rightarrow \mathsf{Tet}(c)$ | $\mathsf{Cube}(b) \rightarrow \neg\mathsf{Tet}(c)$ | 1262 | [INSERT $\neg$ at $\langle.1\rangle$, DELETE $\neg$ at $\langle.2\rangle$] |

Table 1: Some representative error types

we categorize this relabeling as the appropriate subcategory above. A relabeling which involves labels from more than one syntactic category—for example, a constant being replaced by a connective—is called RELABEL$_{Mixed}$. The RELABEL$_{Mixed}$ edit is very rare in the corpus; in fact there are only three homogeneous RELABEL$_{Mixed}$ edit sequences, each of which accounts for one error instance. We note that a similar subtyping of the INSERT and DELETE is possible, but we have not yet made use of this granularity.

Example 1 in Table 1 presents an instance of RELABEL$_{Conn}$, where the student wrote $\vee$ when $\wedge$ was expected; Example 2 in that table presents an instance of RELABEL$_{Const}$, where the student wrote d where a was expected; and Example 3 in the table presents an example of RELABEL$_{Pred}$, where the student wrote SameCol in place of SameRow.

Having identified these different classes of edit operations, we can distinguish two kinds of edit sequences: those that are **homogeneous** and those that are **heterogeneous**. We call an edit sequence **homogeneous** if all of the edit operations in the sequence are of the same type: for example, they might be all INSERT or all RELABEL$_{Conn}$ operations. Any edit sequence that is not homogeneous is **heterogeneous**.

Homogeneous edit sequences are significant from our point of view because they have the potential to be identified and analysed more easily. For example, a homogeneous edit sequence that contains only RELABEL$_{Const}$ edits indicates that the student has made a mistake concerning only the constants and their positions within the answer sentence, but the other features of the sentence are correct. The analysis of these errors based on the information in the edit sequences that they produce is likely to be more tractable, while the analysis of heterogeneous edit sequences, by contrast, will be more involved.

Moreover, homogeneous edit sequences account for more than 60% of the error instances in the corpus. Focussing on these edit sequences enables us to make significant headway in understanding the nature of the errors that students make without having to interpret convoluted edit sequences.

Within the homogeneous edits, we distinguish two categories: **relabellings**, and **insertions and deletions**. A homogeneous relabeling edit consists only of relabelling operations of a single type (for example, all RELABEL$_{Const}$ or all RELABEL$_{Conn}$), indicating that the structure of the formula tree that the student produced was identical to that which was expected (as in Figure 3 shown earlier). Homogeneous insertion and deletion sequences, on the other hand, consist of only INSERT or DELETE operations (although with possibly different operands), meaning that the structure of the formula tree produced by the student was different from that which was expected (Figure 4 shows a case where a single DELETE has occurred).

## 4.1 Relabellings

### *4.1.1* RELABEL$_{Conn}$

The largest category of homogeneous relabellings in the data are sequences of one or more RELABEL$_{Conn}$ edits. These account for 25.10% of the error instances, with 23.45% of the total data set being units; this indicates that students often get the overall structure of the solution correct, but are confused about which logical connectives to use to capture the intended meaning. A simple example of a pair of sentences exhibiting this pattern is the student sentence $\mathsf{Tet(a)} \wedge \mathsf{Tet(b)}$ being provided when the correct answer is $\mathsf{Tet(a)} \rightarrow \mathsf{Tet(b)}$; as shown in Figure 3, here the connective relabeling occurs at the root of the tree.

It is interesting to look more closely at the particular relabellings that students use. Table 2 presents a confusion matrix that shows, for each pair of connectives, how often one connective is found when the other is expected. The numbers here indicate the proportion of attempts that generated the error in each case; so, for example, the pair $\langle$Found: $\leftrightarrow$, Expected: $\rightarrow\rangle$ has a value of 10.33%, indicating that in 10.33% of the situations where a biconditional was expected, the student instead used an implication.[4] Example 4 in Figure 2 provides an example of this extremely common error type.

The other notable confusion here is the use of conjunction when disjunction is required; this occurs in 2.74% of possible cases. For both these confusions, the inverse substitution is also quite prominent, but in neither case to the same extent. This data provides insights into the particular connectives that cause problems for students, and suggest where in the data more detailed analysis might be performed to build an understanding of student misconceptions.

[4]Note that these numbers are for homogeneous relabellings only, i.e., cases where the structure of the formula provided is otherwise correct. It is highly likely that this same error occurs as part of more complex heterogeneous edit sequences, but these are harder to isolate.

| found ⇓ | expected ∧ | ∨ | → | ↔ |
|---|---|---|---|---|
| ∧ | – | 2.74% | 0.50% | 0.44% |
| ∨ | 0.79% | – | 0.07% | 0.06% |
| → | 0.45% | 0.09% | – | 10.33% |
| ↔ | 0.20% | 0.09% | 2.54% | – |

Table 2: Confusion of connectives

| found ⇓ | expected a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | - | 0.11% | 0.20% | 0.07% | 0.10% | 0.31% |
| b | 0.05% | - | 0.15% | 0.47% | 0.48% | 0.04% |
| c | 0.05% | 0.13% | - | 0.12% | 0.42% | 0.02% |
| d | 0.20% | 0.08% | 0.09% | - | 0.13% | 0.05% |
| e | 0.07% | 0.05% | 0.10% | 0.08% | - | 0.08% |
| f | 0.02% | 0.01% | 0.01% | 0.04% | 0.02% | - |

Table 3: Confusion of constants

#### *4.1.2* RELABEL$_{Const}$

Our results show that 13.20% of the error instances are accounted for by homogenous RELABEL$_{Const}$ edits, of which the first 3.32% are units. A confusion matrix illustrating students' constant substitutions is provided in Table 3.

In [3], we reported data from a smaller-scale study of constant substitutions based on data from just one of the LPL exercises. We noted that this kind of error seemed to interact with (1) the use of the constant a in a sentence; (2) whether a was the first-mentioned constant in the sentence; and (3) whether the constant names were alphabetically adjacent (e.g. ⟨a, b, c⟩) as opposed to being alphabetically 'gappy' (e.g. ⟨b, e, d⟩). What we referred to as the **gappiness effect** was statistically significant, being magnified when (1) the letters used as constants were not alphabetically adjacent; and (2) the first constant name mentioned in the sentence was a. Table 3 shows that the most common constant substitution is ⟨Found: b, Expected: e⟩. The NL sentence with which this substitution is most frequently associated is *Neither* **e** *nor* **a** *is to the right of* **c** *and to the left of* **b**. The next two most common substitution patterns are ⟨Found: c, Expected: e⟩ and ⟨Found: b, Expected: d⟩. These three frequent substitutions are ones in which the substituted constant appears earlier in the alphabetic sequence than the expected constant; this is consistent with the operation of a gappiness effect.

Table 3 shows that the fourth largest proportion is associated with ⟨Found: a, Expected: f⟩. An inspection of the NL sentences suggests that this is associated with cases where they contain a and another constant, b. This particular substitution is rarer when a is present as the only other constant, or when the other constants are not a and b (e.g., c and d). This apparent trend warrants further investigation. The fifth most frequent substitution is ⟨Found: d, Expected: a⟩. One sentence (**a** *is to the left or right of* **d** *only if it's a cube*) stands out in relation to this error. Here there is anaphoric reference to an object that is a cube; the intention is that the cube should be labelled a, but clearly many students resolve *it* to the cube with identity d.

#### *4.1.3* RELABEL$_{Pred}$

The RELABEL$_{Pred}$ operation appears relatively infrequently in the data corpus. Only 4.51% of the error instances are accounted for by homogenous RELABEL$_{Pred}$ edits, of which the first 3.32% are units.

An interesting example of this is where the correct sentence is $\mathsf{SameCol}(\mathsf{b},\mathsf{e}) \wedge \mathsf{SameRow}(\mathsf{e},\mathsf{d}) \wedge \mathsf{SameCol}(\mathsf{d},\mathsf{a})$ while the incorrect answer is $\mathsf{SameCol}(\mathsf{b},\mathsf{e}) \wedge \mathsf{SameCol}(\mathsf{e},\mathsf{d}) \wedge \mathsf{SameCol}(\mathsf{d},\mathsf{a})$. This is the second most frequent homogeneous RELABEL$_{Pred}$ sequence, accounting for 769 error instances. Like gappiness, this appears to be the result of students' inability to adopt a sufficiently careful stance toward the formalization task.

In this case, one predicate from the expected language (here, the Blocks World language) is replaced by another from the same language;[5] however, we also find cases where a predicate is replaced by one from outside the language. In the most common instance of this type, instead of the expected answer $2:00 < 2:05$ we find the answer $\mathsf{Earlier}(2:00, 2:05)$: The student has here used a predicate derived from the lexical content of the NL sentence, but this predicate does not exist in the expected formal language (the Pets World language). This error type accounts for 1976 error instances.[6]

## 4.2 Insertions and Deletions

So far we have discussed only relabeling operations, but the tree edit distance computation also allows for the inclusion of INSERT and DELETE operations in the edit sequences. Each such edit operation inserts or deletes a node into the source tree as it is transformed to the target. Unlike relabeling operations, INSERT and DELETE modify the *structure* of the trees in order to align them. Just as a homogeneous relabeling edit sequence may relabel nodes at various points in the tree, a homogeneous sequence of DELETE operations could delete many nodes from the tree. However, we might expect that deletions would come in groups. For example, consider again the error in Example 2 of Figure 2, but this time computing the distance from the correct answer shown for Example 1 in that table. A simple description of this pair is that the incorrect answer has the antecedent and consequent of the implication switched. The shortest edit sequence that achieves this is one which: (1) deletes the node labelled Tet from its position in the consequent (one operation); (2) inserts it as the new antecedent (one operation); (3) removes the node labelled a from its original position (one operation); and (4) inserts a as a child of Tet in its new location, for a total edit distance of 4. But notice that here the INSERT and DELETE operations come in pairs, and that the deleted nodes form a subtree of the original tree. We call these **local** insert or delete sequences. We believe that these local sequences may be of particular interest, because they appear to aggregate as higher level operations.

There is an interesting asymmetry in our data. There are many more homogeneous edit sequences involving INSERT (15.51% of all error instances, with the first 4.34% being units) than there are using DELETE (only 2.16% of all error instances, with the first 1.10% being units). This indicates that students are more likely to provide translations

[5] Most exercises in LPL use the language of a Blocks World; there are also exercises that make use of a 'Pets World' of pets and their owners.

[6] Note that the notion of being outside the language also includes the possibility that a student commits a simple typing error, such as typing Cub where Cube is expected; see Example 4 in Table 1.

that omit information present in the NL sentence than they are to add information. This seems plausible, since any additional information would have to come from somewhere, and this would have to be from outside the source sentence. It is much more likely that students misread the sentence, failing to notice or express relevant information that is present.

One example of students providing 'extra' information in their answer is in an exercise containing the sentence *Notice that f (the large dodecahedron in the back) is not in front of a. Use your first sentence to say this.* In this exercise, students are looking at a picture of a situation in which the sentences are true. We intend the student to translate *f is not in front of a*, and provide a parenthetical hint to help them identify f in the picture. 687 students additionally provide the information that f is a large dodecahedron. This finding suggests that some students are so focussed on the operations required by the translation task that their awareness of problem context is diminished, a process akin to the phenomenon of **suspension of sense-making** proposed in [15].[7]

## 5. HETEROGENEOUS EDITS

As noted above, homogeneous edit sequences account for just over 60% of the data; consequently, 40% of the error instances are heteregeneous, i.e. they consist of more than one type of edit operation.

Our intuition is that particular subsequences (or, more likely, particular subsets) of the edits that make up these edit sequences will correspond to characteristic errors. Identifying these patterns would then allow us to detect the same or similar errors occurring in quite different contexts. However, we observe that, as with so many pattern recognition problems, getting the representation right is crucial, and it is quite likely that the particular set of edit operations we have chosen to use is not optimal for this purpose.

As a case in point: in our earlier work, we identified antecedent–consequent reversal as a particularly prevalent error. From the perspective of tree manipulation, this corresponds to a swapping of two subtrees. Such an operation is only indirectly perceptible in terms of our current edit operations. Although we could derive a characterisation of this error in terms of the lower-level operations, an alternative would be to 'reify' it in terms of higher-level tree manipulation operations; in particular, we might make tree insertion and deletion be primitive operations in determining edit sequences and the corresponding edit distances. The technical challenge here is to retain the tractability of the edit distance computation if these more complex operations are permitted.

As an exploration of the potential of such higher-level edit operations, we observe that one characteristic of the relocation of a subtree is that it will consist of $n$ DELETE operations paired with $n$ INSERT operations.[8] It turns out that 4.94% of the error instances in our data set consist of $n$ INSERTs and $n$ DELETEs, which we might think of as the simplest form of heterogeneous edits. Note that this category does not include longer edit sequences that contain equal numbers of INSERTs and DELETEs in conjunction with other edits, so the actual number of such pairs is likely to be larger.[9]

| DELETE | INSERT | Count |
|---|---|---|
| ¬ | ¬ | 8581 |
| folly | folly | 1406 |
| a | a | 410 |

Figure 7: The three most common symbol movements

One must be careful in interpreting such numbers. It is certainly not the case, for example, that all of these correspond to subtree movements: apart from the fact that other constraints must be met for a subtree movement to be in evidence, the reality is that the vast majority of these paired INSERTs and DELETEs consist of exactly one instance of each, and so they correspond to movements of single symbols rather than larger structures: such movements account for 4.58% of the error instances in the corpus. But even this simpler observation is an important finding: Figure 7 shows the three most common symbol movements that occur in the data set. This reveals that by far the most common error of this type is misplaced negation, as in Example 5 in Table 1; this is clearly a place where students need assistance. The other two cases shown in this table are most likely to be instances of argument movement (although it is conceivable that there are other causes of the phenomena), which arises whenever a student provides an answer that has the arguments in the wrong order.

## 6. CONCLUSIONS

Students find formalisation hard; but the ability to formalise problems underpins many fields, and so any means we can find to improve student abilities in this regard is important and valuable. Our ultimate goal is to develop evidence-based pedagogy in this area, and so we want to take advantage of our large data set of student translations to guide this activity.

Unfortunately, there are no established analytical tools for exploring this kind of data. In this paper, we have presented the results of an initial exploration that uses the notions of edit distance and edit operation sequence as a means to better understanding what it is that students do wrong when they make mistakes in NL to FOL translation. We have been encouraged by the outcomes:

1. Unit errors and edit distance 2 errors account for approx 67% of our data, permitting tractable use of tree edit distance as a useful data mining method in our domain.

2. Results from the application of tree edit distance validate our earlier, pilot findings from analyses of a subsample of the data analysed in this paper. In particular, the analysis provides support for our earlier

[7]As a result of finding this phenomenon in the data, we have subsequently configured the Grade Grinder to accept this as a correct answer.

[8]There are, of course, a number of other constraints, most notably the relationship between the addresses of the nodes in each pair of INSERTs and DELETEs.

[9]We say only likely because it is correspondingly more difficult to determine whether the INSERTs and DELETEs are related to each other; in more complex sequences, it is also possible that some INSERTs or DELETEs belong to some other higher-level transformation.

observations in regard to gappiness in constant substitutions, and antecedent–consequent reversal [3]. Validating previous results by triangulation using different methods promotes confidence in the findings.

As noted throughout this paper, a number of more specific findings provide important stepping-off points for developing a better understanding of the nature of common student errors.

Of course, much remains to be done. So far, we have shown that the techniques described here allow us to identify particular errors that manifest themselves in many different circumstances; but this is really a *descriptive* analysis of the data, and does not necessarily correspond to the kinds of *explanations* for these errors that we might need if we want to provide appropriate assistance to students. In some cases the explanation may seem to follow straightforwardly from the description, but in other cases we need to dig deeper into the context (what are the characteristics of the NL sentence being translated? what is the student's history of error?) in order to provide an explanation. Moving in this direction is a major priority in our future work.

Orthogonal to this line of development, we should note also that we have focussed here on what we call homogenous edits, these being the simpler to identify and analyse. As noted earlier, these account for 60% of the data, so they allow us to make considerable headway; but this leaves the 40% of the data that correspond to heterogeneous edits still to be explored. One strategy we aim to pursue here is to see whether we can decompose heterogeneous edit sequences into partitions, some of which may correspond to the homogenous edit sequences we have already explored; this would then leave a simpler residue still to be characterised. However, we suspect that there may be more mileage to be gained by revising the set of edit operations used to be closer to the kinds of operations that are more conceptually valid in the domain; for example, allowing insertion and deletion of entire subtrees as atomic operations may provide a better way of characterising some of the mistakes that students make.

Armed with these kinds of tools, we hope to be able to develop a better understanding of the mistakes that students make, and to provide appropriate assistance.

## 7. ADDITIONAL AUTHORS

Additional authors: Alex Romanczuk, CSLI/Stanford University, email: `aproman@stanford.edu`.

## 8. REFERENCES

[1] D. Barker-Plummer, R. Cox, and R. Dale. Dimensions of difficulty in translating natural language into first order logic. In *Second International Conference on Educational Data Mining*, pages 220–229. Cordoba Spain, 2009.

[2] D. Barker-Plummer, R. Cox, and R. Dale. Student Translations of Natural Language into Logic: The Grade Grinder Corpus Release 1.0. In *Proceedings of The 4th International Conference on Educational Data Mining*, pages 51–60, 6th–8th July 2011.

[3] D. Barker-Plummer, R. Cox, R. Dale, and J. Etchemendy. An empirical study of errors in translating natural language into logic. In V. Sloutsky, B. Love, and K. McRae, editors, *Proceedings of the 30th Annual Cognitive Science Society Conference*, pages 505–510. Lawrence Erlbaum Associates, 2008.

[4] D. Barker-Plummer, R. Dale, and R. Cox. Impedance effects of visual and spatial content upon language-to-logic translation accuracy. In C. Hoelscher, T. F. Shipley, and L. Carlson, editors, *Proceedings of the 32nd Annual Cognitive Science Society Conference*, pages 3259–3264. Lawrence Erlbaum Associates, 2011.

[5] T. Barnes, J. Stamper, L. Lehmann, and M. Croy. A pilot study on logic proof tutoring using hints generated from historical student data. In T. B. R. Baker and J. Beck, editors, *Proceedings of the 1st International Conference on Educational Data Mining (EDM 2008)*, pages 197–201. International Educational Data Mining Society, 2008.

[6] J. Barwise, J. Etchemendy, G. Allwein, D. Barker-Plummer, and A. Liu. *Language, Proof and Logic*. CSLI Publications and University of Chicago Press, September 1999.

[7] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337:217–239, 2003.

[8] J. Clement, J. Lochhead, and G. Monk. Translation difficulties in learning mathematics. *The American Mathematical Monthly*, 88(4):286–290, 1981.

[9] R. Cox, R. Dale, J. Etchemendy, and D. Barker-Plummer. Graphical revelations: Comparing students' translation errors in graphics and logic. In G. Stapleton, J. Howse, and J. Lee, editors, *Proceedings of the Fifth International Conference on the Theory and Application of Diagrams*. Lecture Notes in Computer Science LNAI 5223, Berlin: Springer Verlag, 2008.

[10] F. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[11] P. N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms (ESA)*, pages 91–102. Springer-Verlag, 1998.

[12] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[13] D. Pimm. *Speaking Mathematically*. Routledge and Kegan Paul, 1987.

[14] J. Reason. *Human Error*. Cambridge University Press, 1990.

[15] L. Verschaffel, B. Greer, and E. de Corte. *Making Sense of Word Problems*. Swets and Zeitlinger, 2000.

# SHORT PAPERS

# Calculating Probabilistic Distance to Solution in a Complex Problem Solving Domain

Leigh Ann Sudol
Carnegie Mellon University
Pittsburgh, PA
leighann@cmu.edu

Kelly Rivers
Carnegie Mellon University
Pittsburgh, PA
krivers@andrew.cmu.edu

Thomas K. Harris
Tutor Technologies
Pittsburgh, PA
thomas@tutortechnologies.com

## ABSTRACT

In complex problem solving domains, correct solutions are often comprised of a combination of individual components. Students usually go through several attempts, each attempt reflecting an individual solution state that can be observed during practice. Classic metrics to measure student performance over time rely on counting the number of submissions or focusing on time taken to complete the problem correctly. These metrics are not robust to the correction of errors that may increase problem solving time, and do not reflect topical misunderstandings on the part of the student. In this paper we propose a metric to measure the probabilistic distance between an observed student solution and a correct solution. Students working in an online programming environment completed four practice problems. Their submissions were then evaluated against a model of the algorithmic components necessary for a correct solution. A Markov Model was used to generate a problem state graph. Our proposed Probabilistic Distance to Solution (PDS) metric was applied to the graph to determine the distance, in program states, from an observed program model to the model of a correct solution. Results indicate that the PDS is useful in determining if an edit or student path is (a) typical of students who have mastered content, and (b) productive in progressing toward a solution. We offer implementation details of PDS and implications for future work based upon current observations.

## 1. INTRODUCTION

Modern data mining and classification techniques allow for increasingly complex solution spaces to be automatically modeled and assessed. For example, automated essay grading[10], mathematical proofs[1], and even complex computer programs[3] can be analyzed for completeness and scored. In these complex problem spaces, novices will often attempt several unique edits and approaches in order to create a finished correct solution. Intelligent Tutoring Systems (ITS) can be used to provide feedback for these attempts based on the scoring criteria used for the assessment. Although the models, feedback strategies and mechanisms used by each domain vary, it is still important for researchers to assess students' progress and make comparisons between research conditions in order to refine and improve such tutoring systems.

In complex problem solving spaces, such as natural language production or computer programming, students may make edits or submit attempts that are not directly related to the specific learning outcomes of the tutoring task[4]. For example, in computer programming, a student may struggle with a compilation error, such as having a parenthesis out of place, which is not reflective of their understanding of the desired learning goal. Students may also produce submissions that progress through multiple skills, creating a complex path to solution, with many possible states[8]. We propose that normal indicators of student performance within tutoring activities, such as time to completion or number of submissions, are too coarse-grained to distinguish between conceptual misunderstanding and syntactical or parsing mistakes that take time and multiple submissions to debug and correct.

In this paper we propose a new metric, Probabilistic Distance to Solution (PDS) and describe its implementation in assessing student progress on an introductory programming assignment. We then apply this metric to a dataset and highlight cases where PDS offers additional insight into misconceptions and problem solving paths.

## 2. PERFORMANCE METRICS

Within-tutor measures of performance are sometimes used instead of running pretests and posttests outside of the tutor, when the creator of the tutor wants more immediate learning feedback. Some within-tutor metrics have already been created and used effectively in tutors; for example, number of submissions and amount of time taken to get to a correct state were used in a system focused on improving math scores[2]. These metrics are not as effective in complex problem solving domains, however, due to the variety of strategies used to solve problems and the difficulty of merging them[5].

Other Intelligent Tutoring Systems use constraint-based modeling to determine how well a program matches up to the expectations of the problem; for example, Mitrovic built an ITS for SQL that used over six hundred constraints to provide accurate and useful hints to students[7]. Le and Menzel also describe techniques for building constraint-based tutors in 'ill-defined domains', similar to the complex domains we describe[6]. However, both of these approaches require that the author of the ITS generate the constraints by hand, which becomes very time-consuming when applied to a broad domain (such as programming). The metric we propose aims to improve on these models by examining more fine-grained aspects of the problem states in a potentially

automatable way[1].

## 3. THE DATA

During the fall of 2011 and winter of 2012, eighteen participants solved four programming problems in an online tutor[9] for computer science. The problems were presented in the same order for each student so that comparison's between students' performances on problem X could be made without the confound of which problems preceded it. Each submission recorded data on the program's text, start and finish time, whether the attempt was successful, and the feedback that was given. Participants' submissions were then evaluated under the model described above and were coded appropriately. Participants generated 354 submissions with 63 observed distinct model states.

### 3.1 Feature Space

For each program submission, the participant's code was translated to a vector of binary features representing the inclusion of semantic program features, correctness outcomes, and compilation success. The semantic program features focuses on the inclusion of algorithmic components such as a looping structure, decision structure, use of the loop control variable in an array access, and inclusion of a return statement. The data collected represents a model of programming as a time series within a high dimensional binary space. The algorithms required by the four problems were similar in terms of features required. This model of required algorithmic elements was based upon the thesis proposal of the first author, and employs identifying the use of semantic structures as well as correctness testing using JUnit.

### 3.2 Two similar solutions

Figure 1 illustrates two participants' paths from an empty start state to a correct finished state. State A represents code that has all of the correct algorithmic components, is compilable, but does not return the appropriate value. State B represents code that has all of the correct algorithmic components, but does not compile, so it cannot check the final return state. Participant 5 corrected a sign error (step 2), then a syntax error (step 3), and finally another sign error (step 4), which resulted in a correct solution. Participant 12, on the other hand, initially submitted code that did not contain a return statement, indicating a misunderstanding about how information is communicated back from the function. State C represents an observed model state where three of the features are marked incorrect for the submission. The participant then made a small change resulting in the same model state for the code (step 2), then added a return statement based on a compile message (step 3), and finally fixed another compile error and submitted a correct solution (step 4). Although these two participants have the same number of submissions, the reasons for and the nature of the submissions are very different and expose a misconception by Participant 12.

Because each submit may represent multiple edits or steps in the problem solving process, simply counting the number of submits as a measure of errors across steps is not informative enough to express the difference between students who make errors with regard to the learning goals of the activity,

[1]Automation of this model's generation will be done in upcoming work, as proposed in [9].

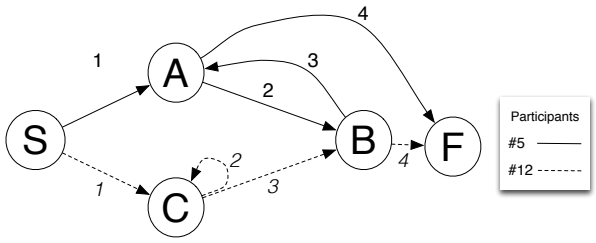



Figure 1: Comparing two student paths - Problem 2

and errors that do not inform measures of desired learning outcomes.

### 3.3 Traditional Measures of Performance

Students tended to reduce overall time taken to solve problems as they moved through the set of problems (see Table 1, where SD is the standard deviation for the indicated problem). Students were performing think-aloud protocols while completing the problems, making the time to submit slightly exaggerated due to verbalization.

| Problem # | | Mean/Median | Min | Max | SD |
|---|---|---|---|---|---|
| # of Submits | 1 | 4.06 / 3 | 1 | 14 | 3.13 |
| | 2 | 7.44 / 4 | 1 | 49 | 11.35 |
| | 3 | 4.56 / 2 | 1 | 23 | 5.71 |
| | 4 | 3.61 / 3 | 1 | 9 | 2.45 |
| Overall Time in Seconds | 1 | 625 / 452 | 93 | 2121 | 510 |
| | 2 | 571 / 364 | 134 | 1795 | 490 |
| | 3 | 437 / 331 | 103 | 1438 | 349 |
| | 4 | 363 / 315 | 53 | 1199 | 279 |

Table 1: Traditional Statistics for Student Data

The ability level of individual participants varied greatly, with some participants submitting final solutions with minimal modifications from their first attempt, while other participants struggled and progressed through multiple incorrect model states before arriving at a correct solution. Individual participants were consistent in their performance across problems either doing well or struggling with all of them. The traditional measurement metrics can be used to separate participants into two groups: high performers (students who were able to quickly solve the problems), and low performers (students who needed time and several attempts to get a problem right).

Of the seven students requiring more than six submissions to solve at least one problem, only two averaged fewer than six submissions per problem. These seven students also tended to take more than 500 seconds (8.33 minutes) overall to solve their problems, apart from the two mentioned above, who have individual outliers above that line. This disjoint grouping suggests that we can subdivide the low performing group into students who performed uniformly badly and students who struggled only with a specific problem.

The eleven high performers all averaged four or fewer submissions to reach a correct answer, and all clustered under an average of 400 seconds (6.66 minutes), with the exception of one student who took nearly eighteen minutes to finish the first problem, but only needed to submit once.

# 4. PROBABILISTIC DISTANCE TO SOLUTION

In order to draw generalizations about how program states correspond to student performance and other latent factors such as learning, we aggregated all student submission paths for each problem into a network (see Figure 2). The network nodes $S_1 \cdots S_{n-1}$ are possible program states with an end state node $E$, and the edges are the observed transitions between states.

For each node, we use our observations to compute a Maximum Liklihood Estimate (MLE) transition probabilities to every other node. Given the number of observed transitions from state $x$ to state $y$ ($T_{x,y}$), we estimate the probability of being in state $y$ at time $t$, with the MLE:

$$\hat{p}(S_y(t)) = P(S_y(t)|S_x(t-1)) = \frac{T_{x,y}}{\sum_i T_{x,i}} \quad (1)$$

This is a equivalent to a Markov chain estimate with a 1-state history.[2]

By modeling each edge as a transition probability and distance between states, we use a set of linear equations to calculate a mean distance from each state to the end (successful completion) state. With

- $n-1$ non-terminal states $S_1 \cdots S_{n-1}$ and an end state $E$,
- and with each state $S$ having transition probabilities $P_{s,1} \cdots P_{s,n-1}$ and $P_{s,e}$,
- and transition distances $D_{s,1} \cdots D_{s,n-1}$ and $D_{s,e}$,

a system of equations for the mean distance to the end state $d_e(S)$ is:

$$\begin{aligned}
d_e(S_1) &= \left[\sum_{s=1}^{n-1} P_{1,s}(D_{1,s} + d_e(S_s))\right] + P_{1,e}D_{1,e} && (2)\\
d_e(S_2) &= \left[\sum_{s=1}^{n-1} P_{2,s}(D_{2,s} + d_e(S_s))\right] + P_{2,e}D_{2,e} && (3)\\
&\vdots && (4)\\
d_e(S_{n-1}) &= \left[\sum_{s=1}^{n-1} P_{n-1,s}(D_{n-1,s} + d_e(S_s))\right] \\
&\quad + P_{n-1,e}D_{n-1,e} && (5)\\
d_e(E) &= 0 && (6)
\end{aligned}$$

For the case where we are interested only in the mean number of *submissions* to the end state, each $D_{x,y} = 1$, and the calculation simplifies to the system of dot products:

$$\begin{aligned}
d_e(S_1) &= \vec{\boldsymbol{P_1}} \bullet d_e(\vec{\boldsymbol{S}}) + 1 && (7)\\
d_e(S_2) &= \vec{\boldsymbol{P_2}} \bullet d_e(\vec{\boldsymbol{S}}) + 1 && (8)\\
&\vdots && (9)\\
d_e(S_{n-1}) &= \vec{\boldsymbol{P_{n-1}}} \bullet d_e(\vec{\boldsymbol{S}}) + 1 && (10)\\
d_e(E) &= 0 && (11)
\end{aligned}$$

There are several variations on the algorithm that we would like to explore, given a larger dataset with more statical power. In particular, these variations include treating state-features differently, so that transitions between states that differ in particular features (e.g., compilability) will be able to indicate different latent variables of the students, (e.g., example concept understanding).

# 5. APPLYING PDS

The PDS metrics accompanied by the transition graph are rich sources of information about the paths that participants pursued in order to arrive at a correct solution. Figure 2 includes a table illustrating the observed model states in the binary vector, as well as the PDS for each state. In problem 4, S56 is the solution state and S0 the initial starting state. Before even evaluating the student paths, we can observe that an additional state, S63, was also a terminal state for a participant. This participant located a bug in the evaluation system that has since been corrected.[3]

By looking at the PDS combined with the Program State Graph (PSG) we can identify more productive edits by participants. For example one participant's first submission was observed as S55 (PDS 2.99), and next state was S57 (PDS 4.43). This edit would be less productive as it resulted in a transition to a state with a greater probabilistic number of submits required to obtain a correct solution.

With the possibility of including terms in the algorithm for syntactic but not model changes (i.e. two states that are identical except for a compilation error would not be counted as a full step), PDS can be adapted to focus on model state transitions that indicate misconceptions or other model-based goals of the data miner.

# 6. CONCLUSIONS

Within these early results, we have already identified model states on productive and unproductive PDS paths. Using the actual PDS values we can determine if a student is making a productive edit, engaging in either guessing behavior, or pursuing a misconception. An edit resulting in an observed state with a higher PDS than the prior submission indicates a move away from a correct answer.

These values can be invaluable to tutor designers as they seek to develop feedback and support tools for complex solution domains. Within computer programming tutors, the PDS could offer implications for more-than-compiler support, and perhaps even prompt the introduction of a similar worked example or code comprehension problem highlighting the incorrect features of the model.

[2]We believe that the student's state transitions will be better represented by a higher-order Markov process; however our current data set is too small to provide appropriate power for more than a first-order analysis.

[3]The bug was identified as a part of the think aloud protocol, however the PDS and Student Program States Graph would have identified the bug for the tutor designers as well.

| | F2 | F4 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | P(distance) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | | | | | | | | | | | | 3.67 |
| S19 | | | | | | | | | | | | 7.78 |
| S35 | | | | | | | | | | | | 4.78 |
| S54 | | | | | | | | | | | | 4.00 |
| S55 | | | | | | | | | | | | 2.99 |
| S56 | | | | | | | | | | | | 0.18 |
| S57 | | | | | | | | | | | | 4.43 |
| S58 | | | | | | | | | | | | 3.00 |
| S59 | | | | | | | | | | | | 4.78 |
| S60 | | | | | | | | | | | | 2.17 |
| S61 | | | | | | | | | | | | 6.78 |
| S62 | | | | | | | | | | | | 3.09 |
| S63 | | | | | | | | | | | | -0.00 |

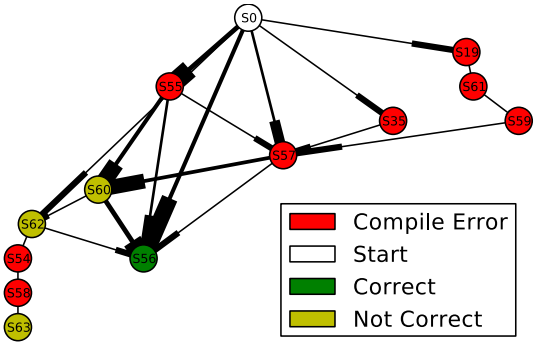



Figure 2: Student program states for problem 4. Table columns Fx are binary program features. Table rows Sx are observed combinations of those features in program submissions. Only observed states (nodes) and transitions (edges) are shown. Node self-transitions also exist in the model, but are not shown here. The thicknesses of the edges are proportional to the log of observed transitions in the data. The lengths of the edges are arbitrary and do not relate to the model.

Although tested against data from a computer programming dataset, the authors believe that the PDS metric could be valuable across many domains with complex solutions demonstrating multiple skills. Future work is planned to use the PDS metric on a larger dataset to extract common paths and evaluate differences between tutoring conditions.

## 7. ACKNOWLEDGEMENTS

This work was supported through the Program for Interdisciplinary Education Research (PIER) at Carnegie Mellon University, funded through Grant R305B040063 to Carnegie Mellon University, from the Institute of Education Sciences, US Department of Education. The opinions expressed are those of the authors and do not represent the views of the Institute or the US Department of Education. We would also like to thank Dr. Christy McGuire of Tutor Technologies for her assistance in preparing this manuscript.

## 8. REFERENCES

[1] Tiffany Barnes and John C. Stamper. Automatic Hint Generation for Logic Proof Tutoring Using Historical Data. *Educational Technology & Society*, 13(1):3–12, 2010.

[2] Mingyu Feng, Neil T. Heffernan, and Kenneth R. Koedinger. Predicting State Test Scores Better with Intelligent Tutoring Systems: Developing Metrics to Measure Assistance Required. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, pages 31–40, 2006.

[3] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of Recent Systems for Automatic Assessment of Programming Assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pages 86–93, 2010.

[4] Matthew C. Jadud. A First Look at Novice Compilation Behaviour Using BlueJ. *Computer Science Education*, 15(1):25–40, 2005.

[5] H. Chad Lane and Kurt VanLehn. Intention-Based Scoring: An Approach to Measuring Success at Solving the Composition Problem. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 373–377, 2005.

[6] Nguyen-Thinh Le and Wolfgang Menzel. Using Constraint-Based Modelling to Describe the Solution Space of Ill-defined Problems in Logic Programming. In *Advances in Web Based Learning (ICSL 2007)*, pages 367–379, 2007.

[7] Antonija Mitrovic. An Intelligent SQL Tutor on the Web. *International Journal of Artificial Intelligence in Education*, 13(2-4):173–197, 2003.

[8] James Spohrer, Elliot Soloway, and Edgar Pope. A goal/plan analysis of buggy pascal programs. *Human Computer Interaction*, 1:463–207, 1985.

[9] Leigh Ann Sudol. Deepening Students' Understanding of Algorithms: Effects of Problem Context and Feedback Regarding Algorithmic Abstraction. *Carnegie Mellon Thesis Proposal*, 2011.

[10] Salvatore Valenti, Francesca Neri, and Alessandro Cucchiarelli. An Overview of Current Research on Automated Essay Grading. *Journal of Information Technology Education*, 2:319–330, 2003.

# Classification via clustering for predicting final marks based on student participation in forums

M.I. López, J.M Luna, C. Romero, S. Ventura
Department of Computer Science and Numerical Analysis
University of Córdoba
Córdoba, Spain
i32loqum@uco.es, i32luarj@uco.es, cromero@uco.es, sventura@uco.es

## ABSTRACT

This paper proposes a classification via clustering approach to predict the final marks in a university course on the basis of forum data. The objective is twofold: to determine if student participation in the course forum can be a good predictor of the final marks for the course and to examine whether the proposed classification via clustering approach can obtain similar accuracy to traditional classification algorithms. Experiments were carried out using real data from first-year university students. Several clustering algorithms using the proposed approach were compared with traditional classification algorithms in predicting whether students pass or fail the course on the basis of their Moodle forum usage data. The results show that the Expectation-Maximisation (EM) clustering algorithm yields results similar to those of the best classification algorithms, especially when using only a group of selected attributes. Finally, the centroids of the EM clusters are described to show the relationship between the two clusters and the two classes of students.

## Keywords

classification via clustering, prediction, classification, social networks analysis, forums

## 1. INTRODUCTION

Forums have recently become one of the leading means of peer communication on the internet. An internet forum is a web application for publishing user-generated content in the form of a discussion. Internet forums are sometimes called web forums, discussion boards, message boards, discussion groups, or bulletin boards [10]. The most important feature of internet forums is their social aspect. Many forums are active for a long period of time and attract a group of dedicated users, who build a tight social community within the forum. These social aspects of a discussion can highlight user interest in a specific topic. Current research activities use data mining to discover this information, especially in educational contexts, where online discussion forums are the best way to share ideas, post problems, comment on posts by other students, and obtain feedback [13]. In fact, mining group activities in a learning context provides quantifiable group profiles, which allow us to (1) evaluate the collaborative activity that the participants carry out, (2) analyse the link structure of the group, (3) compare the collaborative performance of different groups, and (4) predict behaviours and reveal link patterns [6] and collaboration trends. Mining data generated by students communicating using forum-like tools can help reveal aspects of their communication [14]; for example, the more students participate in the forum for a certain course, the more involved they will be in the subject matter of that course. Following this line, in this study we try to test whether or not there is a correlation between the participation of students in Moodle [4] forums and their final course marks. We have developed a new and specific Moodle module in order to obtain directly both statistics and social network information based on student forum usage data. We also propose the use of a classification via clustering approach to predict the final marks on the basis of our forum dataset.

The rest of the paper is organised as follows: a short theoretical background is presented in Section 2, the proposed methodology is outlined in Section 3, Section 4 describes the forum data used, Section 5 presents the experimental results, and conclusions and future research are outlined in Section 6.

## 2. BACKGROUND

Forums are one of the most commonly used tools in web-based teaching-learning environments because they play an important role in students' collaborative learning [12]. In fact, student activity in discussion threads can be a relevant source of information that facilitates the monitoring of tasks during the course by providing teachers with relevant indicators of student needs and weaknesses [3]. The use of data mining is a potential strategy for discovering and building alternative representations for the data underlying discussion forums [5]. The literature encourages analysis of forum interactions to reveal student characteristics and behaviour [1]; however, there is less published work on the use of data mining to predict student performance based on forum usage data. Classification is one of the oldest and most useful data mining tasks used to predict student outcomes, marks, or scores [15], and some works have used all the tracking data provided by Learning Management Systems (LMSs) in relation to visits and times, resources viewed, assessments, and activities in chat rooms, forums, etc. [2],[16]. However, the use of clustering for classification has not yet been applied in an educational context. Although clustering is normally an unsupervised process for grouping similar elements (students in this case) into clusters, classification can be performed based on clustering if we use the class information to evaluate the obtained clusters. This approach has been used to develop an anomaly-based network intrusion detection system [11], to predict heart disease in medical diagnosis [7], and to develop an effective system for classification of multidimensional data via clustering. [9]. However, we have found no work that uses only forum-usage data to predict final marks or that uses a classification via clustering approach in an educational context.

## 3. PROPOSED APPROACH

In this work, we propose to use a meta-classifier that uses a cluster for classification approach based on the assumption that each cluster corresponds to a class (see Figure 1). Firstly, the usage and interaction forum data have to be collected and

preprocessed. Then, an optional attribute selection process can be applied (B), or not (A), in order to select only a group of attributes/variables or to use all available. Next, a clustering algorithm is executed using the training data, after removal of the class attribute, and the mapping between classes and clusters is determined. This mapping is then used to predict class labels for unseen instances in test data. In other words, the class attribute is not used in clustering, but it is used to evaluate the obtained clusters as classifiers.

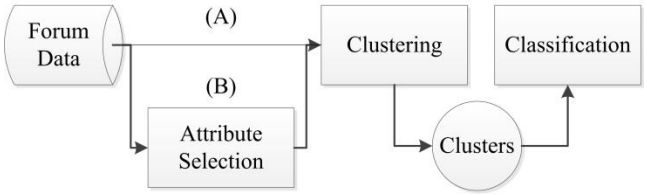



Figure 1: Proposed classification via clustering approach

For all cluster algorithms, it is important to ensure that the number of clusters generated is the same as the number of class labels in the dataset in order to obtain a useful model that relates each cluster with one class. We use this approach to test if student participation in forums is related to whether they pass or fail the course.

## 4. DESCRIPTION OF THE DATA USED

The dataset used in this work was gathered from a Moodle forum used by university students during a first-year course in computer engineering in 2011 (see Table 1).

| Number of students | Number of messages | Number of threads | Number of replies |
|---|---|---|---|
| 114 | 1014 | 81 | 933 |

Table 1: Some forum statistics

We developed a new module for Moodle specifically to obtain a summary dataset file with basic forum usage statistics (see Figure 2), to perform some analysis of social networks, to facilitate teacher evaluation of the messages, and to add the final marks of the students.

| Student | nMessages | nThreads | nReplies | nWords | nSentences | nR |
|---|---|---|---|---|---|---|
| [redacted] | 3 | 0 | 3 | 67 | 3 | |
| [redacted] | 6 | 1 | 5 | 513 | 1 | |
| [redacted] | 1 | 1 | 0 | 17 | 2 | |
| [redacted] | 2 | 0 | 2 | 43 | 2 | |

Figure 2: Screenshot of Moodle forum module

This tool not only enables us to visualise a list of variables for each student (see Table 2) but also allows us to save this summary information in a PDF file for report purposes or in an Excel file for data mining purposes.

| Attribute | Description |
|---|---|
| nMessages | Number of messages sent per student |
| nThreads | Number of threads created per student |
| nReplies | Number of replies sent per student |
| nWords | Number of words written by student |
| nSentences | Number of sentences written by student |
| nReads | Number of messages read on the forum |
| tTime | Total time, in hours, spent on forum |
| aEvaluation | Average score of the messages |
| dCentrality | Degree centrality of the student |
| dPrestige | Degree prestige of the student |
| fMark | Final mark obtained by the student |

Table 2: Variables of a student in a forum

The variables relating to forum usage are nMessages, nThreads, nReplies, nWords, nSentences, nReads, and tTime. The variable aEvaluation is the average score of the messages sent by the student. This evaluation of the contextual meaning of the messages has been done manually by the course teacher, who has read all the messages and assigned a score between 0 (bad) and 3 (very good). The two social network analysis measures are dCentrality and dPrestige, which are closely related to hyperlink analysis [8]. Both centrality and prestige are measures of the degree of prominence of an actor in a social network. Central or prominent actors are those that are extensively linked or involved with other actors. A person with extensive contacts (links) or communications with many other people in the organisation is considered more important than a person with relatively fewer contacts. Prestige is a more refined measure of the prominence of an actor than centrality. A prestigious actor is defined as one who is the recipient of extensive ties.

Finally, the class or attribute to be predicted in this study is fMark, that is, the final mark obtained in the final exam at the end of the course. It has two possible values or labels: PASS or FAIL.

## 5. EXPERIMENTAL RESULTS

All our experiments were performed using Weka [17] and the previously described forum dataset. In order to test the accuracy of obtained classification models we used the 10-fold cross-validation method. All classifiers in Weka work in the same way under cross-validation. The model is built using just the instances in the training fold. The classification via clustering approach is based on the "clusters to classes" evaluation routine in the cluster evaluation code, which finds a minimum-error mapping of clusters to classes.

In the first experiment, we executed the following clustering algorithms provided by Weka for classification via clustering using all the available attributes (see Table 2): EM, FarthestFirst, HierarchicalClusterer, sIB, SimpleKMeans, and XMeans.

In the second experiment, we repeated all the previous executions using fewer attributes, based on the assumption that not all the available attributes are discriminative factors in the final marks. A process of feature selection was used to identify which attributes could have the greatest effect on our class (final mark). Weka provides a range of feature-selection algorithms from which we selected ten:CfsSubsetEval, ChiSquaredAttributeEval, ConsistencySubset-Eval, FilteredAttributeEval, FilteredSubsetEval, GainRatio-AttributeEval, InfoGainAttributeEval, OneRAttributeEval, ReliefFAttributeEval, and SVMAttributeEval. To rank the attributes, we counted the

number of times each attribute was selected by each attribute-selection algorithm (see Table 3). Finally, we selected as the best attributes the first six attributes in the ranking, because these were selected by at least half (5) of the algorithms.

| Attribute | Frequency |
|---|---|
| dCentrality | 9 |
| nMessages | 8 |
| nReplies, nWords | 7 |
| dPrestige | 6 |
| aEvaluation | 5 |
| nSentences, nReads, nThreads | 3 |
| tTime | 1 |

Table 3: Attributes ranked by frequency of appearance

The previous clustering algorithms were then executed for classification via clustering but using only the six selected attributes (see Table 3, above the bold line). Table 4 shows the overall accuracy (rate of correctly classified students) using all the available attributes (A) and using only the six selected attributes (B).

| Clustering algorithm | (A) | (B) |
|---|---|---|
| EM | **0.842** | **0.894** |
| FarthestFirst | 0.526 | 0.535 |
| HierarchicalClusterer | 0.578 | 0.570 |
| sIB | 0.710 | 0.578 |
| SimpleKMeans | 0.666 | 0.640 |
| Xmeans | 0.666 | 0.640 |

Table 4: Accuracy of classification via clustering approach

An analysis of the results shown in Table 4 reveals that only one algorithm obtained a good level of accuracy. In fact, the EM algorithm obtained the highest accuracy in both cases (A and B) and the best overall accuracy (89.4%) when using only the six selected attributes. All the other clustering algorithms obtained much worse accuracy values (50%–70%) than EM, and, in general, there was no improvement by using only six attributes.

In the third experiment, we compared the accuracy of the previous classification via clustering approach with that of traditional classification algorithms by executing a representative number of classifications of different types:

- Rules-based algorithms: DTNB, JRip, NNge, and Ridor
- Trees-based algorithms: ADTree, J48, LADTree, and RandomForest
- Functions-based algorithms: Logistic, MultilayerPerceptron, RBFNetwork, and SMO
- Bayes-based algorithms: BayesNet and NaiveBayesSimple

Table 5 shows the accuracy obtained by the previous classification algorithms using all the attributes (A) and only the six selected attributes (B).

| Algorithms | (A) | (B) |
|---|---|---|
| DTNB | 0.859 | 0.833 |
| JRip | 0.833 | 0.815 |
| NNge | 0.842 | 0.807 |
| Ridor | 0.833 | 0.842 |
| ADTree | 0.859 | 0.842 |
| J48 | 0.824 | 0.807 |
| LADTree | 0.868 | 0.850 |
| RandomForest | 0.850 | 0.833 |
| Logistic | 0.859 | 0.850 |
| MultilayerPerceptron | 0.842 | 0.868 |
| RBFNetwork | 0.868 | 0.886 |
| SMO | 0.868 | 0.886 |
| BayesNet | **0.877** | 0.842 |
| NaiveBayesSimple | 0.859 | **0.894** |

Table 5: Accuracy of classification algorithms

All the algorithms obtained a good accuracy with more similar values (80%–90%) than those obtained previously by the classification via clustering approach. The results indicate that some algorithms improve when using only six attributes, but others do not. The highest results are obtained by BayesNet when using all the attributes (87.7%) and NaiveBayesSimple when using only six attributes (89.4%), which is the best overall accuracy and is equal to that obtained by the EM algorithm.

Finally, we show the cluster centroids for the EM algorithm when using the six selected attributes that have yielded the best accuracy (see Table 6). The clusters-to-classes mapping done by the EM algorithm is such that cluster 0 is mapped to FAIL class and cluster 1 is mapped to PASS class.

| Attributes | Cluster 0 | Cluster 1 |
|---|---|---|
| nMessages | 1.2199 | 14.8905 |
| nReplies | 1.1599 | 13.6718 |
| nWords | 18.4599 | 668.8039 |
| aEvaluation | 0 | 0.7751 |
| dCentrality | 0.0011 | 0.1565 |
| dPrestige | 0 | 0.1021 |

Table 6: Cluster centroids obtained by EM algorithm

Cluster centroids describe the typical student for each group or cluster (see Table 6). We can see that the obtained clusters can be very informative from the point of view of classifying good and bad students. In fact, students who show a great level of participation in the forum (cluster 1) are classified as PASS, and students who show a very low level of participation in the forum (cluster 0) are classified as FAIL.

## 6. CONCLUSIONS

This paper demonstrates the potential of the classification via clustering approach in an educational context, using it to predict students' final marks on the basis of their participation in forums.

Based on the results obtained using several clustering and classification algorithms, we can answer the two initial questions:

a) Yes, student participation in the course forum was a good predictor of the final marks for the course. Another advantage of classification models based on mapping clusters to classes is that they are very simple and interpretable to instructors. In the case presented here, instructors only have to analyse the cluster centroids to know that students active in the forum pass the course and passive students fail.

b) Yes, the proposed classification via clustering approach obtained similar accuracy to traditional classification algorithms using our forum data. However, our proposed approach only had to obtain a good accuracy when using the EM algorithm (compared with traditional classification algorithms). On the other hand, the feature selection process can be useful to in reducing the number of attributes without losing reliability in classification. However, although some algorithms improved their classification performance when using only the selected attributes, the accuracy of other algorithms decreased.

However, in order to generalise the result obtained, the experiments must be repeated using different forum data to test if the same results are obtained or not, that is, if the EM clustering algorithm obtains again a high accuracy comparable with traditional classification algorithms. In the future, we hope to automate the process of evaluating student messages, because evaluating messages manually is a very difficult and time-consuming task for instructors. A data text mining algorithm could be used to automatically detect and classify types of messages and evaluate them. Finally, we are working on improving our Moodle forum module. We hope to develop a network analysis tool to graphically depict the forum interaction (sociograms) and to identify further measures than the two currently used (centrality and prestige) to provide valuable information for predicting students' final marks.

## 7. ACKNOWLEDGMENTS

This work was supported by the Regional Government of Andalusia and the Spanish Ministry of Science and Technology projects, P08-TIC-3720 and TIN-2011-22408, respectively, FEDER funds and the Spanish Ministry of Education under the FPU grant AP2010-0041.

## 8. REFERENCES

[1] A. R. Anaya and J. G. Boticario. Content-free collaborative learning modeling using data mining. *User Modeling and User-Adapted Interaction*, pp. 1–36. Springer, 2011.

[2] M. Calvo-Flores, E. Galindo, and M. Jiménez. Predicting students' marks from Moodle logs using neural network models. *Current Developments in Technology-Assisted Education*, 1:586–590, 2006.

[3] G. Cobo, D. García, E. Santamaría, J. A. Morán, J. Melenchón, and C. Monzo. Modeling students' activity in online discussion forums: a strategy based on time series and agglomerative hierarchical clustering. *Proceedings of Educational Data Mining*, 253-258, 2011.

[4] J. Cole and H. Foster. *Using Moodle: Teaching with the popular open source course management system.* O'Reilly Media, Inc., 2007.

[5] L.P. Dringus and T. Ellis. Using data mining as a strategy for assessing asynchronous discussion forums. *Computers & Education*, 45(1):141–160, 2005.

[6] L. Getoor. Link mining: a new data mining challenge. *ACM SIGKDD Explorations Newsletter*, 5(1):84–89, 2003.

[7] S. Jyoti, A. Ujma, S. Dipesh, and S. Sunita. Predictive data mining for medical diagnosis: An overview of heart disease prediction. *International Journal of Computer Applications*, 17(8):43–48, 2011.

[8] N. Memon, J.J. Xu, D.L. Hicks and H. Chen. Data Mining for Social Network Data. 1-8, Springer 2010.

[9] R. Krakovsky and R. Forgac. Neural network approach to multidimensional data classification via clustering. *Intelligent Systems and Informatics (SISY), 2011 IEEE 9th International Symposium on*, 169–174, IEEE2011.

[10] M. Morzy. On mining and social role discovery in internet forums. *Social Informatics, 2009. SOCINFO'09. International Workshop*, 74–79. IEEE, 2009.

[11] M. Panda and M. Patra. A novel classification via clustering method for anomaly based network intrusion detection system. *International Journal of Recent Trends in Engineering*, 2:1–6, 2009.

[12] R. Rabbany, M. Takaffoli and O. Zaïane. Analyzing participation of students in online courses using social network analysis techniques. *Proceedings of Educational Data Mining*, 21-30, 2011.

[13] P. Raghavan, R. Catherine, S. Ikbal, N. Kambhatla, and D. Majumdar. Extracting problem and resolution information from online discussion forums. *Management of Data*, 77, 2010.

[14] P. Reyes and P. Tchounikine, Mining learning groups' activities in forum-type tools. *Proceedings of the 2005 conference on computer support for collaborative learning: Learning 2005: the next 10 years!* 509–513. International Society of the Learning Sciences, 2005.

[15] C. Romero, S. Ventura, P. Espejo, and C. Hervás. Data mining algorithms to classify students. *Proceedings of Educational Data Mining*, 20-21, 2008.

[16] C. Romero, S. Ventura and E. García. Data mining in course management systems: Moodle case study and tutorial. *Computers & Education*, 51(1):368–384, 2008.

[17] I.H. Witten, F. Eibe and M.A. Hall. Data Mining, Practical Machine Learning Tools and Techniques. Third Edition. Morgan Kaufman Publishers, 2001.

# Development of a Workbench to Address the Educational Data Mining Bottleneck

Ma. Mercedes T. Rodrigo
Ateneo de Manila University
Loyola Heights, Quezon City, Philippines
mrodrigo@ateneo.edu

Ryan S. J. d. Baker
Worcester Polytechnic Institute
Worcester, MA USA
rsbaker@wpi.edu

Bruce M. McLaren
Carnegie Mellon University
Pittsburgh, PA USA
bmclaren@cmu.edu

Alejandra Jayme
Ateneo de Manila University
Loyola Heights, Quezon City, Philippines
alejandra.jayme@gmail.com

Thomas T. Dy
Ateneo de Manila University
Loyola Heights, Quezon City, Philippines
thatsmydoing@gmail.com

## ABSTRACT

In recent years, machine-learning software packages have made it easier for educational data mining researchers to create real-time detectors of cognitive skill as well as of metacognitive and motivational behavior that can be used to improve student learning. However, there remain challenges to overcome for these methods to become available to the wider educational research and practice communities, including developing the labels that support supervised learning, distilling relevant and appropriate data features, and setting up appropriate cross-validation and configuration algorithms. We discuss the development of an Educational Data Mining (EDM) Workbench designed to address these challenges.

## Keywords

Educational data mining workbench

## 1. INTRODUCTION

In recent years, educational data mining methods have afforded the development of detectors of a range of constructs of educational importance, from gaming the system [5] to off-task behavior [3] to motivation [8] to collaboration and argumentation moves [11]. The development of these detectors has been supported by the availability of machine learning packages such as RapidMiner [12], WEKA [15], and KEEL [1]. These packages provide large numbers of algorithms of general use, reducing the need for implementing algorithms locally, however they do not provide algorithms specialized for educational data mining, such as the widely used Bayesian Knowledge-Tracing [7]. Furthermore, effective use of these packages by the educational research and practice communities presumes that key steps in the educational data mining process have already been completed. For example, many of these detectors have been developed using supervised learning methods, which require that labeled instances, indicative of the categories of interest, be provided. Typically, many labeled instances – on the order of hundreds, if not thousands – are required to create a reliable behavior detector. Labeling data is a time consuming and laborious task, made even more difficult by the lack of tools available to support it.

A second challenge is the engineering and distillation of relevant and appropriate data features for use in detector development [15]. The data that is directly available from log files typically lacks key information needed for optimal machine-learned models. For instance, the gaming detectors of both [5] and [14] rely upon assessments of how much faster or slower a specific action is than the average across all students on a problem step, as well as assessments of the probability that the student knew the cognitive skills used in the current problem step. This information can be distilled and/or calculated by processing data across an entire log file corpus, but there are currently no standard tools to accomplish this. Feature distillation is time-consuming, and many times a research group re-uses the same feature set and feature distillation software across several projects (the second author, for instance, has been using variants of the same feature set within Cognitive Tutors for nine years). Developing appropriate features can be a major challenge to new entrants in this research area. To address this "data labeling bottleneck" and the difficulty in distilling relevant features for machine learning, we are working to develop an *Educational Data Mining (EDM) Workbench*. A beta version of this Workbench, now available online at http://penoy.admu.edu.ph/~alls/downloads, is described in this paper. The workbench currently allows learning scientists to

1) label previously collected educational log data with behavior categories of interest (e.g. gaming the system, help avoidance), considerably faster than is possible through previous live observation or existing data labeling methods.
2) collaborate with others in labeling data.
3) automatically distill additional information from log files for use in machine learning, such as estimates of student knowledge and context about student response time (i.e. how much faster or slower was the student's action than the average for that problem step).

Through the use of this tool, we hope that the process of developing a detector of relevant metacognitive, motivational, engagement, or collaborative behaviors can eventually be sped up. Just the use of "text replays", on previously collected log data has been shown to speed a key phase of detector development by about 40 times, with no reduction in detector goodness [5].

## 2. EDM WORKBENCH

Version 1.0 of the EDM Workbench interfaces with some of the tools discussed in Section 1, filling some of the functional gaps that, without the Workbench, require manual intervention or require hand-coding of custom tools and cumbersome and complex actions by the user in packages such as Excel. Version 1.0 of the Workbench has five functionalities: Log import, feature

distillation, data sampling, data clipping and labeling, and data export. We discuss each of these functions in turn.

## 2.1 Log import

The EDM Workbench allows users to import logs in DataShop text format [9] and CSV. The data is assumed to be stored in a flat file, organized in rows and columns. The first row of the import file is assumed to contain each column's name. Each succeeding row represents one logged transaction, usually between the student and tutor but possibly between two or more students as in the case of collaborative learning scenarios. If the user specifies that the imported data is in DataShop text format, the Workbench will check whether the table contains the columns it requires to distill 26 pre-defined features (discussed in 2.2). The successfully-imported logs may be saved in the Workbench's format for work files—a compressed file containing the data in CSV format plus metadata specific to the EDM Workbench.

The Workbench can also import nested folders of data, where each folder level represents a meaningful subset of the data. For example, if data from a section of students is collected several times over a school year, the researcher may have one folder for the school year, one subfolder for each section within the school year, one subfolder for a session within each section, and finally one file or folder for each student within a session. The Workbench allows users to label each level of subfolder, creating new columns for these labels, appending them to the data tables during importation process.

## 2.2 Feature distillation

Assuming the necessary columns exist in the imported file, the Workbench can automatically distill 26 features from the data. The Workbench also has capacity for defining new features for future analyses. The 26 features distilled come from features used in past automated behavior detectors using DataShop data and related intelligent tutoring system data [2, 5, 13, 14]. The features include (but are not limited to) estimates of the student's knowledge of the current skill [7]; the time the student spent on the problem (both in absolute and relative terms); and the types, number and proportion of correct, wrong, or help actions for the current skill for the last n steps, for the skill, or for the student.

The current EDM Workbench uses 21 generic functions to compute the 26 automatically distilled features. Some functions correspond directly to a single feature while others are reusable, i.e. users can vary input parameters to compute for different features. Figure 1 is an excerpt of the EDM Workbench configuration file that specifies the features to be distilled and the functions used to distill them. It shows the specification of two features: `timeSD` and `timelastnSD`. These features have been used in several behavior detectors [2, 3].

The first example in the excerpt is the specification for the feature `timeSD`, which makes use of a generic function also named `timeSD`. `<group_col>` refers to a sub-grouping criterion. In this case, the data is grouped by type of step, as specified in the `Step Name` column. The `<range_col>` is the column that contains the duration values that will be used to compute `timeSD`. Finally, `<out>` specifies the feature and output column name.

The second example in the excerpt is the specification for `timelastnSD`. It uses the function `sumLastN`. `<sort_col>` refers to the column by which the data should be sorted before computing the feature. The two sets of `<group_col>`s imply that data sub-grouping in this case is based on two criteria, the `Anon Student Id` and the `Problem Name`. The `<range_col>` refers to the `timeSD` column, computed earlier. The `<n>` refers to the number of steps to be used in the computation. As with the first example, `<out>` specifies the feature and output column name.

**Figure 1. Excerpt from the EDM Workbench configuration file.**

```
<feature_set>
 <timeSD>
  <group_col>Step Name</group_col>
  <range_col>Duration</range_col>
  <out>timeSD</out>
 </timeSD>
 <sumLastN>
  <sort_col>Row</sort_col>
  <group_col>Anon Student Id</group_col>
  <group_col>Problem Name</group_col>
  <range_col>timeSD</range_col>
  <n>3</n>
  <out>timelastnSD</out>
 </sumLastN>
</feature_set>
```

Feature `timeSD`

Feature `timelastnSD`

At the moment, adding new features for distillation requires some programming: If the feature can be computed by using one of the 21 existing functions, the user can modify the EDM configuration file to define the new function and how it is derived. If a feature requires a new function, the user can add the new function to the EDM Workbench's source code, after which the new feature can be defined in the configuration file. It is our long-term objective to foster a user community that will eventually make new features available for others to use, similar to the open source software community, increasing the EDM Workbench's usefulness to the broader research community.

## 2.3 Clip generation

In different projects, text replays have been implemented in several different ways [5, 10, 13]. Two of the key ways that text replays have differed has been in terms of the information and grain-size of the data presented to the coder. For coding, data is subdivided into smaller units, termed *clips* — subsets of student-tutor transactions defined based on criteria for when they begin and end, and what information is included. For example, in various projects, clips have been defined as 20-second intervals [5], segments of 5 or 8 actions [10], and in terms of defined "begin" and "end" events in the learning software [13].

The EDM Workbench allows the user to define the set of features by which the data should be grouped, so that clips do not contain rows from different groups. For example, if the data should be grouped by student, a single clip will contain data from only one student and not multiple students. The workbench also specifies the clip size, either by time or by number of transactions. Delineation of clips by beginning and ending events is not yet possible, but is a feature planned for future implementation. The Workbench then generates the clips for analysis, according to a sampling scheme discussed in the next section.

## 2.4 Data sampling

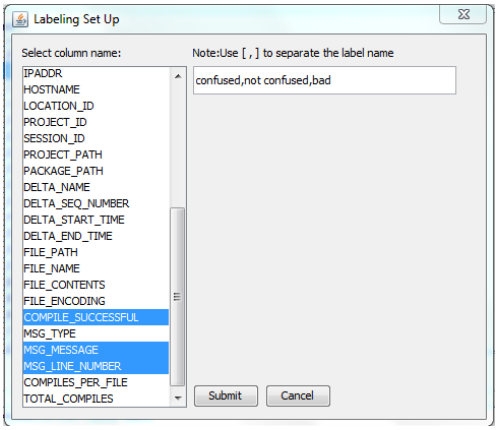
The data sampling feature of the Workbench allows the user to specify how clips are sampled from the data set. (It can also be used to sample at the action/transaction level). The user can specify the sample size, and whether the Workbench will randomly take the sample across the entire population or whether the workbench will stratify the sampling based on one or more variables.

Note that the Workbench allows the user to sample the data at any point of the process — after importing, after clipping, or after labeling – depending on the user's analytical goals.

## 2.5 Labeling

Once the sample has been taken, the user must then specify a subset of the clip columns that should be displayed in the text replay. It is possible that the user does not want all the clip columns displayed in the text replay. In the example shown in Figure 2, the user specified that only three columns will be displayed: COMPILE_SUCCESSFUL, MSG_MESSAGE and MSG_LINE_NUMBER. The user also specifies the labels that the observer or expert will use to characterize each clip. Figure 3 (bottom) shows that expert or observer will have three labels to choose from: Confused, Not Confused or Bad Clip – the coding scheme from [10]. The circumstances under which an expert or observer labels a clip as "bad" changes depending on the data set, but typically indicate cases that should not be coded. For example, in the case of [10], a clip was labeled "bad" if the transactions contained instructor-supplied programming examples rather than programs that the students had written themselves.

**Figure 2. Specification of clip columns and labels.**



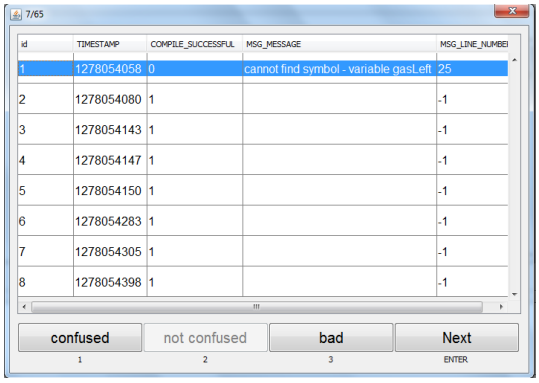

The Workbench then displays text replays of the clips together with the labeling options (Figure 3). A coder reads through the text replay and selects the label that best describes the clip. The labels are saved under a new column in the data set.

Because a coder may have to label tens of thousands of clips [5], the coder may save his or her work and can continue the labeling process in a later session.

## 2.6 Feature distillation and export

Once data labeling is complete, the user can create clip-level data features to associate with the clips, facilitating later development of detectors. The user first selects the feature or column of interest. The user then specifies whether he/she would like the Workbench to compute for the minimum, maximum, average or standard deviation of that feature [13]. The Workbench will add the new column and corresponding computed value results to the clip dataset

**Figure 3. Text replay and label options.**




Finally, once processing is complete, the Workbench allows the user to save the logs in CSV format, for re-importation into an appropriate data mining tool, such as RapidMiner or WEKA. The user is then able to use that tool to build a detector of the construct they labeled, using the features they distilled.

# 3. FUTURE WORK

In this paper, we have presented the Educational Data Mining Workbench, a tool that researchers can use to facilitate the development of detectors of varying forms of student behavior. Version 1.0 of the Workbench supports two key steps of the detector development process that are relatively difficult and time-consuming to do with existing tools: data labeling and feature distillation. By scaffolding users in conducting either or both of these steps, the tool may make it easier and quicker for a wider range of learning scientists and educational software developers to develop and use automated detectors of student behavior.

It is worth noting that the current version of the Workbench is still limited. Each of the limitations discussed here are scheduled for implementation in the coming months. (1) The automatically-distilled features are hard-coded; future releases will make it easier to alter the feature list. (2) The process of amending XML to create new features will be made more user-friendly. (3) The coders cannot change the way in which the text replays are displayed; future releases will support configuration of different ways to pretty print the text replays, towards highlighting the most important information for the coder's specific current purpose. (4) Users can currently only sample data and assemble it into clips in a limited number of fashions; we intend to implement more sophisticated sampling and clip-creation strategies [13].

A final direction for future work is to add support for researchers creating and validating models appropriately. Within the educational data mining community, there has emerged considerable know-how about how to set up tools such as RapidMiner to afford appropriate validation. (For example, batching data in order to support k-fold student-level cross-validation, and then using a BatchXValidation operator in RapidMiner to implement it). We plan to add support for automatically creating appropriately stratified batches to realize several common cross-validation strategies, and automatically export RapidMiner code that is set up to read in the correct data and use appropriate cross-validation to build a detector of the construct that was labeled.

Development of the EDM Workbench remains ongoing, and we look forward to collaborating with a range of EDM researchers and learning scientists in making this tool as useful as possible for the EDM community. We welcome comments and suggestions – as well as contributions – from any interested colleague.

## 4. ACKNOWLEDGMENTS

We thank Jessica Sugay, Alipio Gabriel, and John Paul Contillo. We also thank John Stamper, Alida Skogsholm, and Ken Koedinger for helpful comments and suggestions. This research project was made possible through a grant from the Philippines Department of Science and Technology's Engineering Research and Technology for Development program entitled "Development of an Educational Data Mining Workbench."

## 5. REFERENCES

[1] Alcala-Fdez, J., Sanchez, L., Garcia, S., de Jesus, M.J., Ventura, S., Garrell, J. M., Otero, J., Romero, C., Bacardit, J. & Rivas, V.M. (2009). KEEL: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing: A Fusion of Foundations, Methodologies and Applications, 13*(3), 307-318.

[2] Baker, R.S.J.d., Corbett, A.T., Aleven, V. (2008). More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 406-415.

[3] Baker, R.S.J.d. (2007). Modeling and Understanding Students' Off-Task Behavior in Intelligent Tutoring Systems. *Proceedings of ACM CHI 2007: Computer-Human Interaction*, 1059-1068.

[4] Baker, R.S.J.d., Corbett, A.T., Koedinger, K.R., & Wagner, A.Z. (2004). Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game the System." *Proceedings of ACM CHI: Computer-Human Interaction*383-390.

[5] Baker, R.S.J.d. & de Carvalho (2008). Labeling Student Behavior Faster and More Precisely with Text Replays. *1st International Conference on Educational Data Mining*, 38-47.

[6] Cetintas, S., Luo, S., Yan Ping Xin, & Hord, C. (2010). Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques. *IEEE Transactions on Leanring Technologies*, *3*(3), 228-236.

[7] Corbett, A.T., & Anderson, J.R. (1995). Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction, 4*, 253-278.

[8] de Vicente, A., Pain, H. (2002). Informing the detection of the students' motivational state: an empirical study. *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, 933-943.

[9] Koedinger, K., Cunningham, K., Skogsholm, A., Leber, B. (2011) An data repository for the EDM community: The PSLC DataShop. In c. Romero, S. Ventura, M. Pechenizkiy and R. S. J. d. Baker, *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press, 43-55.

[10] Lee, D. M., Rodrigo, M. M. T. R., Baker, Ryan S. J. D., Sugay, J. O., & Coronel, A. (2011). Exploring the relationship between novice programmer confusion and achievement. In S. D'Mello & A Graesser (Eds.): *ACII 2011, Part I, LNCS 6974*, (pp. 175-184), Berlin Heidelberg: Springer-Verlag.

[11] McLaren, B.M., Scheuer, O., & Mikšátko, J. (2010). Supporting collaborative learning and e-Discussions using artificial intelligence techniques. *International Journal of Artificial Intelligence in Education (IJAIED) 20*(1), 1-46.

[12] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M. & Euler, T. (2006). YALE: Rapid Prototyping for Complex Data Mining Tasks. In *Proc. of the 12th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining (KDD 2006)*, (pp. 935-940), ACM Press.

[13] Sao Pedro, M., Baker, R., Gobert, J., Montalvo, O., & Nakama, A. (in press). Leveraging Machine-Learned Detectors of Systematic Inquiry Behavior to Estimate and Predict Transfer of Inquiry Skill. *User Modeling and User-Adapted Interaction.*

[14] Walonoski, J. & Heffernan, N.T. (2006). Detection and Analysis of Off-Task Gaming Behavior in Intelligent Tutoring Systems. In Ikeda, Ashley & Chan (Eds.). *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. Springer-Verlag: Berlin. pp. 382-391.

[15] Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, Second Edition, Morgan Kaufmann.

# Early Prediction of Student Self-Regulation Strategies by Combining Multiple Models

Jennifer L. Sabourin
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27695
jlrobiso@ncsu.edu

Bradford W. Mott
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27695
bwmott@ncsu.edu

James C. Lester
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27695
lester@ncsu.edu

## ABSTRACT

Self-regulated learning behaviors such as goal setting and monitoring have been found to be crucial to students' success in computer-based learning environments. Consequently, understanding students' self-regulated learning behavior has been the subject of increasing interest. Unfortunately, monitoring these behaviors in real-time has proven challenging. This paper explores a variety of data mining approaches to predicting student self-regulation capabilities. Students are classified into SRL-use categories based on evidence of goal-setting and monitoring activities. Prior work on early prediction of these categories pointed to logistic regression and decision tree models as effective techniques. This paper builds on these findings by exploring techniques by which these models can be combined to improve classification accuracy and early prediction capabilities. By improving classification accuracy, this work can be leveraged in the design of computer-based learning environments to provide adaptive scaffolding of self-regulation behaviors.

## Keywords

Self-regulated learning, early prediction, machine learning

## 1. BACKGROUND

Understanding and facilitating students' self-regulated learning behaviors has been the subject of increasing attention in recent years. This line of investigation is fueled by evidence suggesting the strong role that self-regulatory behaviors play in a student's overall academic success [1]. Self-regulated learning (SRL) can be described as "the process by which students activate and sustain cognitions, behaviors, and affects that are systematically directed toward the attainment of goals" [2]. Unfortunately, students can demonstrate a wide range of fluency in their SRL behaviors [3] with some students lagging behind their peers in their ability to appropriately set and monitor learning goals. Findings that students with low SRL skills are less likely to achieve academic success have prompted efforts to mediate these differences [1,4].

Identifying and scaffolding SRL strategies has also been a focus of much work in the intelligent tutoring systems community. For example, in MetaTutor, a hypermedia environment for learning biology, think-aloud protocols have been used to examine which strategies students use, while analysis of students' navigation through the hypermedia environment helps to identify profiles of self-regulated learners [5]. Similarly, researchers have identified patterns of behavior in the Betty's Brain system that are indicative of low and high levels of self-regulation [6]. Prompting students to use SRL strategies when these patterns of behavior occur has shown promise in improving student learning. For example, Conati et al. have examined the benefits of prompting students to self-explain when learning physics content and how these explanations can be facilitated in a computer-based learning environment [7].

Such work has focused primarily on examining SRL in highly structured problem-solving and learning environments. However, understanding and scaffolding students' SRL behaviors is particularly important in open-ended learning environments where goals may be less clear and students do not necessarily have a clear indicator of their progress [8]. In order to be successful in this type of learning environment, students must actively identify and select their own goals and evaluate their progress accordingly. While the nature of the learning task may have implicit overarching goals such as 'completing the task' or 'learning a lot,' it is important for students to set more specific, concrete and measurable goals [9]. Unfortunately, students do not consistently demonstrate sufficient self-regulatory behaviors during interactions with these environments, which may reduce the educational potential of these systems [10,11]. Consequently, identifying and scaffolding students with low SRL skills is a necessary next step to ensure that these systems can be used as effective learning tools.

This paper reports on an investigation of self-regulatory behaviors of students in a game-based science mystery, CRYSTAL ISLAND. During interactions with the CRYSTAL ISLAND environment, students were prompted to report on their mood and status in a way that is similar to many social networking tools available today. Though students were not explicitly asked about their goals or progress, many students included this information in their short, typed status statements. This data is used to classify students into low, medium, and high self-regulated learning behavior classes. Prior work has pointed to the importance of being able to identify and scaffold the low SRL students [4]. While logistic regression and decision tree models have been found to be effective at early prediction of these classes, this work expands upon these findings by exploring ways in which these models can be combined to improve classification accuracy and early prediction capabilities. Ensemble methods have been found to be effective at a variety of predictive tasks including predicting student knowledge [12]. By improving classification accuracy, this work can be leveraged in future systems to provide adaptive scaffolding of self-regulation behavior early into interaction with the environment, offering the possibility for timely intervention. The implications of these results and areas of future work are then discussed.

## 2. METHOD

The investigation of students' SRL behaviors was conducted with students from a middle school interacting with CRYSTAL ISLAND, a game-based learning environment being developed for the domain of microbiology that follows the standard course of study for eighth grade science in North Carolina.

### 2.1 CRYSTAL ISLAND

CRYSTAL ISLAND features a science mystery set on a recently discovered volcanic island. Students play the role of the protagonist, Alex, who is attempting to discover the identity and source of an unknown disease plaguing a newly established research station. The story opens by introducing the student to the island and the members of the research team for which her father serves as the lead scientist. As members of the research team fall ill, it is her task to discover the cause and the specific source of the outbreak. Typical game play involves navigating the island, manipulating objects, taking notes, viewing posters, operating lab equipment, and talking with non-player characters to gather clues about the disease's source. To progress through the mystery, a student must explore the world and interact with other characters while forming questions, generating hypotheses, collecting data, and testing hypotheses.

### 2.2 Study Procedure

A study with 296 eighth grade students was conducted. After removing instances with incomplete data or logging errors, there were 260 students remaining. Among the remaining students, there were 129 male and 131 female participants varying in age and race. Participants interacted with CRYSTAL ISLAND in their school classroom, although the study was not directly integrated into their regular classroom activities. Pre-study materials were completed during the week prior to interacting with CRYSTAL ISLAND. The pre-study materials included a demographic survey, researcher-generated CRYSTAL ISLAND curriculum test, and several validated instruments. Personality was measured using the Big 5 Personality Questionnaire, which indexes subjects' personalities across five dimensions: openness, conscientiousness, extraversion, agreeableness and neuroticism [12]. Goal orientation was measured using a 2-dimensional taxonomy considering subjects' mastery or performance orientations along with their approach or avoidance tendencies [13]. Subjects' affect regulation tendencies were measured with the Cognitive Emotion Regulation Questionnaire [14] though features from this survey were not included in the current models.

Immediately after solving the mystery, or after 55 minutes of interaction, students moved to a different room in order to complete several post-study questionnaires including the curriculum post-test.

Students' affect data were collected during the learning interactions through self-report prompts. Students were prompted every seven minutes to self-report their current mood and status through an in-game smartphone device. Students selected one emotion from a set of seven options, which included the following: anxious, bored, confused, curious, excited, focused, and frustrated. After selecting an emotion, students were instructed to briefly type a few words about their current status in the game, similarly to how they might update their status in an online social network.

### 2.3 SRL Classification

The typed status reports were later tagged for SRL evidence using the following four ranked classifications: 1) specific reflection, 2) general reflection, 3) non-reflective statement, or 4) unrelated (Table 1). This ranking is motivated by the observation that setting and reflecting upon goals is a hallmark of self-regulatory behavior and that specific goals are more beneficial than those that are more general [9]. Students were then given an overall SRL score based on the average score of their statements. An even tertiary split was then used to assign the students to a Low, Medium, and High SRL category.

**Table 1.** Categories of SRL tags

| SRL Category | Description | Examples |
|---|---|---|
| Specific reflection | Student evaluates progress towards a specific goal or area of knowledge | "I am trying to find the food or drink that caused these people to get sick."<br>"Well...the influenza is looking more and more right. I think I'll try testing for mutagens or pathogens – [I] ruled out carcinogens" |
| General reflection | Student evaluates progress or knowledge but without referencing a particular goal | "I think I'm getting it"<br>"I don't know what to do" |
| Non-reflective | Student describes what they are doing or lists a fact without providing an evaluation | "testing food"<br>"in the lab" |
| Unrelated | Any statement which did not fall into the above three categories is considered unrelated, including non-word or unidentifiable statements | "having fun"<br>"arghhh!" |

From the 260 students, a total of 1836 statements were collected, resulting in an average of 7.2 statements per student. All statements were tagged by one member of the research team with a second member of the research team tagging a randomly selected subset (10%) of the statements to assess the validity of the protocol. Inter-rater reliability was measured at $\kappa = 0.77$, which is an acceptable level of agreement. General reflective statements were the most common (37.2%), followed by unrelated (35.6%), specific reflections (18.3%) and finally non-reflective statements (9.0%).

The tertiary split of students into Low, Medium, and High SRL classes has yielded interesting findings in prior work [4]. One important finding is that Medium and High SRL students have both higher prior knowledge and higher learning gains than Low SRL students. This suggests that Low SRL students begin with some disadvantage and that the overall gap in knowledge is increased after interactions with CRYSTAL ISLAND. Though all groups have significant learning gains, Low SRL students are not experiencing the same advantages of interaction with CRYSTAL ISLAND. This finding points to the strong need to provide these students with additional scaffolding to improve the quality of their interaction.

### 2.4 SRL Prediction

The difference in learning between Low, Medium, and High SRL students has motivated the goal of early prediction of students' SRL skills. Prior work [4] has shown promise in being able to predict SRL class early into the interaction. This work compared the ability of naïve Bayes, neural network, logistic regression,

support vector machine, and decision tree models to predict SRL class at different time intervals. Overall it was found that logistic regression and decision trees offered the best performance, with the best model correctly predicting 57% of students' classes after one-third of their interaction with CRYSTAL ISLAND. Compared with a most-frequent-class baseline of 34%, this offers a significant improvement in the ability to recognize SRL skill. However, while both logistic regression and decision tree models significantly outperformed other modeling techniques, neither of the two best performers consistently outperformed the other. This raised the question of whether some method of combining these two learned models might offer improved or more stable performance.

### *2.4.1 Original Models*

The original logistic regression and decision tree models were trained using 10-fold cross validation with the WEKA machine learning toolkit [15]. For the original models, a total of 49 features were used to train machine-learning models. Of these, 26 features represented personal data collected prior to the student's interaction with CRYSTAL ISLAND. This included demographic information, pre-test score, and scores on the personality, goal orientation, and emotion regulation questionnaires. The remaining 23 features represented a summary of student's interactions in the environments. This included information on how students used each of the curricular resources, how many in-game goals they had completed, as well as evidence of off-task behavior. Additionally, data from the student's self-reports were included, such as the most recent emotion report and the character count of their "status".

In order to examine early prediction of the students' SRL-use categories, these features were calculated at four different points in time resulting in four unique datasets. The first of these (**Initial**) represented information available at the beginning of the student's interaction and consequently only contained the 26 personal attributes. Each of the remaining three datasets (**Report1**-**3**) contained data representing the student's progress at each of the first three emotion self-report instances. These datasets contained the same 26 personal attributes, but the values of the remaining 23 in-game attributes differentially reflected the student's progress up until that point. The first self-report occurred approximately 4 minutes into game play with the second and third reports occurring at 11 minutes and 18 minutes, respectively. The third report occurs after approximately one-third of the total time allotted for interaction has been completed, so it is still fairly early into the interaction time.

### *2.4.2 Combining Multiple Models*

To combine the predictions of multiple models, a variety of different voting schemes were used in which both the predicted class from the original decision tree and logistic regression models were taken into account:

- **Standard**: The prediction from each model is weighted equally.
- **Weighted by Accuracy**: The prediction from each model is weighted by the model's overall predictive accuracy.
- **Weighted by Precision:** The prediction from each model is weighted by its precision at predicting the class for which it is voting.
- **Select Lowest Class**: The model predicting the lowest SRL skill is selected.

The final model of always selecting the lowest level prediction is based on the assumption that we would rather underestimate students' abilities and provide additional scaffolding than overestimate their abilities. Additionally, in all of the above voting schemes, the lower class was chosen in case of a tie.

## 3. RESULTS

For each time slice, we compared the original models with the combined models by evaluating overall predictive accuracy as well as recall on the Low-SRL class. The first metric represents how well the model does overall at correctly identifying each class, while the latter represents the proportion of Low-SRL students who were correctly identified. This second metric is especially important given the proposed style of intervention. These metrics for each model are shown in Table 2.

The results indicate that the most successful voting model was the **Weighted by Precision** model. It offered statistically significantly ($p < 0.05$) better accuracy than any other model, and better Low-SRL recall than either original model for all time-slices, with the exception of the **Initial** prediction. It also offered improved stability of performance over the original models and other ensemble models, with both accuracy and recall improving as more data became available. The **Select Lowest Class** combined model had the highest recall of the Low-SRL class which is to be expected given its favoritism for low classifications. The **Select Lowest Class** model identified almost exactly half of all students as Low-SRL However, it was able to correctly identify up to 85% of the actual Low-SRL students, making it a promising contender for identifying cases where additional scaffolding would be beneficial.

**Table 2.** Predictive models and evaluation metrics

| | ***Predictive Accuracy*** | | | | ***Low-SRL Recall*** | | | |
|---|---|---|---|---|---|---|---|---|
| | Initial | Report1 | Report2 | Report3 | Initial | Report1 | Report2 | Report3 |
| **Original Models** | | | | | | | | |
| Decision Tree | 37.7 | 46.5 | 51.6 | 53.4 | 0.36 | 0.58 | 0.63 | 0.70 |
| Logistic Regression | 40.8 | 55.0 | 53.1 | 57.2 | 0.43 | 0.65 | 0.68 | 0.77 |
| **Combined Models** | | | | | | | | |
| Standard Voting | 38.1 | 50.0 | 54.3 | 54.1 | 0.45 | 0.67 | 0.75 | 0.79 |
| Weighted by Accuracy | 37.1 | 53.1 | 55.0 | 54.5 | 0.33 | 0.56 | 0.65 | 0.65 |
| Weighted by Precision | 40.1 | 57.3 | 57.0 | 59.1 | 0.44 | 0.67 | 0.75 | 0.80 |
| Select Lowest Class | 36.9 | 51.5 | 52.3 | 51.4 | 0.58 | 0.81 | 0.79 | 0.84 |

With the exception of the **Weighted by Precision** model, the predictive accuracy of each ensemble model tended to fall somewhere between accuracy of the original decision tree and logistic regression models. This suggests that these models did not have enough additional information in their weighting scheme to offer improvements in performance. It is especially interesting that weighing votes by overall accuracy was not beneficial. This is likely due to the high and mostly equivalent accuracies of both the original models. However, the **Weighted by Precision** model takes into account each model's likelihood of correctness given a particular prediction which varied between models. Specifically, the logistic regression model was generally better at Low and High SRL predictions while the decision tree model was stronger at Medium SRL predictions.

# 4. CONCLUSION

Predicting students' self-regulated learning skills can form the basis for effective scaffolding strategies. Combining multiple machine learned models can be used for early prediction of students' self-regulated learning skills, as was shown in an investigation with the narrative-centered learning environment, CRYSTAL ISLAND. Results indicate that early prediction of self-regulation skills is feasible and that combining multiple models can offer improvements over individual models alone. Specifically, logistic regression and decision tree models were combined using a variety of voting strategies. Some of these strategies were able to offer significant improvements in both predictive accuracy and Low-SRL recall.

These findings point to several directions for future work. The most prominent of these is developing intervention mechanisms for aiding student self-regulation. Early prediction of SRL skills is not useful unless we are able to act intelligently upon this prediction. Therefore, the development of appropriate and effective scaffolding strategies is an important next step in this line of investigation. These techniques could then be used in conjunction with several of the top-performing models in order to determine which optimizations have the best impacts on students overall learning.

# 5. ACKNOWLEDGMENTS

The authors wish to thank members of the IntelliMedia Group for their assistance, Omer Sturlovich and Pavel Turzo for use of their 3D model libraries, and Valve Software for access to the SourceTM engine and SDK. This research was supported by the National Science Foundation under Grants REC-0632450, DRL-0822200, IIS-0812291, and CNS-0739216. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

# 6. REFERENCES

[1] Zimmerman B.J. Self-regulated learning and academic achievement: An overview, *Educational Psychologist*, vol. 25, pp. 3-17, 1990.

[2] Schunk, D. H. Attributions as Motivators of Self-Regulated Learning, in *Motivation and Self-Regulated Learning: Theory, Research, and Applications*, D. H. Schunk and B. J. Zimmerman (eds). 2008, pp. 245-266.

[3] Ellis, D. and Zimmerman, B. J. Enhancing self-monitoring during self-regulated learning of speech, H. J. Hartman (ed) Dordrecht, The Netherlands: Kluwer, 2001, pp. 205-228.

[4] Sabourin, J. L., Shores, L.R., Mott, B. W., and Lester, J. C. Predicting Student Self-Regulation Strategies in Game-Based Learning, to appear in *Proceedings of the 11th International Conference on Intelligent Tutoring Systems*, 2012.

[5] Azevedo, R., Johnson, A., Chauncey, A., and Burkett, C. Self-Regulated Learning with MetaTutor: Advancing the Science of Learning with MetaCognitive Tools, in *New Science of Learning: Cognition, Computers and Collaboration in Education*, M. Khine and I. Saleh (eds) New York: Springer, 2010, pp. 225-248.

[6] Biswas, G., Jeong, H, Roscoe, R., and Sulcer, B. Promoting Motivation and Self-Regulated Learning Skills through Social Interactions in Agent-Based Learning Environments, in *2009 AAAI Fall Symposium on Cognitive and Metacognitive Educational Systems*, 2009.

[7] Conati, C. and VanLehn, K. Towards Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation, *International Journal of Artificial Intelligence in Education*, vol. 11, pp. 398-415, 2000.

[8] Land, S. Cognitive requirements for learning with open-ended learning environments, *Educational Technology Research and Development*, vol. 48, no. 3, pp. 61-78, 2000.

[9] Zimmerman, B. Goal Setting: A Key Proactive Source of Academic Self-Regulation, in *Motivation and Self-Regulated Learning: Theory, Research, and Applications*, D. H. Schunk and B. J. Zimmerman (eds) New York: Routledge, 2008, pp. 267-286.

[10] Kirschner, P. A., Sweller, J. and Clark, R. E. Why Minimal Guidance during instruction does not work: An analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching, *Educational Psychologist*, vol. 41, pp. 75-86, 2006.

[11] Alfieri, L., Brooks, P., Aldrich, N., and Tenenbaum, H. Does Discovery-Based Instruction Enhance Learning, *Journal of Education Psychology*, vol. 103, pp. 1-18, 2011.

[12] Pardos, Z.A., Gowda, S. M., Baker, R. S.J.D., Heffernan, N. T. The Sum is Greater than the Parts: Ensembling Models of Student Knowledge in Educational Software. To appear in *ACM's Knowledge Discovery and Datamining Explorations*, vol. 13, 2012.

[13] McCrae, R. and Costa, P. *Personality in Adulthood: A Five-Factor Theory Perspective*, 2nd ed. New York: Guilford Press, 1993.

[14] Elliot, A. and. McGregor, H.A . A 2 x 2 achievement goal framework, *Journal of Personality and Social Psychology*, vol. 80, pp. 501-519, 2001.

[15] Gernefski, N. and Kraati, V. Cognitive Emotion Regulation Questionnaire: Development of a Short 18-Item Version, *Personality and Individual Differences*, vol. 41, pp. 1045-1053, 2006.

[16] Hall, M., Frank, E., Holmes, G., Pfajhringer, B., Reutmann, P., and Witten, I. The WEKA Data Mining Software: An Update, *SIGKKD Explorations*, vol. 11, no. 1, 2009.

# Identifying Successful Learners from Interaction Behaviour

Judi McCuaig
School of Computer Science
University of Guelph
Guelph, ON, Canada
judi@uoguelph.ca

Julia Baldwin
School of Computer Science
University of Guelph
Guelph, ON, Canada
baldwinj@uoguelph.ca

## ABSTRACT

The interaction behaviours of successful, high-achieving learners when using a Learning Management System (LMS) are different than the behaviours of learners who are having more difficulty mastering the course material. This paper explores the idea that conventional Learning Management Systems can exploit data mining techniques to predict the success or failure of students without requiring the results of formal assessments. This paper describes a study with a second semester computer science class that shows that the success or failure of a learner can be predicted using information about learner interactions with course materials and learner self-reports of subject matter confidence.

## 1. INTRODUCTION

A Learning Management System gathers and records a rich set of data about the educational materials for a course and about the learners using the course. The data is used by the LMS in producing reports about the learners and the educational materials. Data gathered by a LMS is also frequently used in data mining applications to make predictions about learners. Three common uses for the data are: to predict the learners' likely academic performance and adjust either the LMS or the face to face instruction accordingly, to automatically adapt the LMS or the content to the needs or learner preferences of individual learners, or to predict learner affect in an effort to monitor and react appropriately to learner engagement with the course.

This paper presents a study that shows that successful learners exhibit different interaction behaviours with the LMS than less successful learners. Our results indicate that the results of formal assessments are unnecessary to accurately predict which learners will be successful with the course content and which learners will struggle with the course.

## 2. MINING LMS METADATA

A common goal for mining the metadata produced by learning management systems is to predict learner achievement. Most achievement-predicting systems produce a prediction of final grade (see for example [**?**]). Others predict or model the knowledge that the learner has mastered, with the goal of adapting the materials in the course to more closely match the learner's immediate learning needs (see for example [**?**]). The data mined by such systems almost always include the correctness or score for learning objects that generate grades, elapsed times for completing activities, and the learner interactions with the LMS.

LMS interaction data usually consists of a list of the elements viewed by learners as well as dates and timestamps to identify the viewing. Also, in many LMS the specific types of LMS objects (ie. forums, quizzes, help pages) are identified in the logs, making the type of learning object another piece of metadata that is frequently collected. All of this data can be used in learner classification activities.

## 3. MINING FOR LEARNER SUCCESS

A study was conducted during the winter of 2011 in a single semester computer science class on programming in the C language. 122 learners participated in the study, which captured a wide variety of data for the entire duration of the course. The work presented in this paper looks at the relationship between the learner's activities on the LMS and their ultimate success in the course. For this portion of the study, the learner's final grade was used as the ground truth representation of overall learner success. A ranking of final grades in a single course, regardless of the grade distribution, will most likely result in the more successful learners appearing in the top ranks, and the less successful learners appearing in the bottom ranks.

### 3.1 Data Collection

Throughout the semester experimenters collected information about participant interactions with course material. Data was obtained from the course LMS, from the Subversion (SVN) server, from in-class clickers, and from formal observations.

Data included information about the learners habits with respect to lecture attendance and participation, starting and finishing labs and assignments, and studying from online materials. The resulting data set included the dates and time(s) that participants read a lab or assignment description, dates and times that assignments and labs were submitted, the date and time that a participant had their lab graded (labs were graded in-person), lecture attendance, the dates that problem sets were completed, and the amount of time taken to complete quizzes and exams. Additional data was generated through the collection of self assessment surveys, which asked learners to rate their own confidence and mastery of the course material.

Each week, along with the self-assessment questions, learners were given an ungraded set of multiple choice problems to help them evaluate their understanding of the course con-

tent. The weekly problem set consisted of seven randomly selected questions. Every problem set consisted of 2 hard, 2 medium and 3 easy questions that spanned the entire course content. Learners had the option of indicating that a question was about content they had yet to learn rather than guess at an answer. A measure of the participant's self-reported confidence as well as the participant's success with problem sets was calculated for each week of the course.

Because participation in this study was voluntary, and not all students in the course participated, the data collection process could not interfere with the normal activities of the learners in the course. As a result, the study activities were ungraded and did not affect the academic grade of the learner. One of the side-effects of this restriction was that participants sometimes placed little importance on the study-specific activities (the problem sets and self-evaluation).

## 3.2 Analysis

A goal of this work is to determine if learner success (or failure) can be predicted using data that is passively, or semi-passively captured as the learner works through the course. For the purposes of this work, passively and semi-passively captured data is data that can be collected exclusively from the learner's interactions with the LMS, and without the need for separate action on the part of the course instructor or teaching assistants. In some cases the data may come from log files and in others it may be directly captured from questions asked of the learners.

### *3.2.1 Self Assessments*

Independent self-assessment activities such as ungraded problem sets for self-checking and opportunities for self-reflection about personal confidence and progress are one type of data that can be semi-passively captured. The data capture requires input from the learner, but not from course administration or instructors.

During this study, participants completed weekly problem sets that were kept at the same difficulty for the entire semester. The problem sets were difficult for learners at the beginning of the semester and should have seemed easier as the learners mastered the course material. No problem sets were assigned in weeks 11 and 12. The data collected during the study shows that the median scores on the problem sets slowly increased over the semester. The increase is not dramatic, nor is it consistent every week, but higher median scores did occur in the last third of the semester as can be seen in Table 1.

The participants' weekly scores on the problem sets show a weak correlation with their final grade, however the total problem set score shows a higher correlation ( .73) with final grade (see Figure 1).

Participants also assessed their own confidence in programming skills each week. When examined week-by-week, the correlation between learner confidence and final grade is consistently positive except for a one week anomaly shortly after a difficult quiz. The scores from the weekly confidence assessments were averaged across two separate 5 week periods in the course, weeks 0 through 5 and weeks 5 through 10 (no confidence assessments were done in weeks 11 and 12). The correlation between those mean confidence scores and participant final grade was calculated. The results can be seen in Table 2.

Table 1: Median Scores for Problem Sets by Week

| Week | Mean | Median |
|---|---|---|
| 0 | 3 | 3 |
| 1 | 4 | 4 |
| 2 | 4 | 4 |
| 3 | 4 | 4 |
| 4 | 4 | 5 |
| 5 | 4 | 4 |
| 6 | 4 | 4 |
| 7 | 5 | 5 |
| 8 | 4 | 4 |
| 9 | 5 | 5 |
| 10 | 4 | 5 |

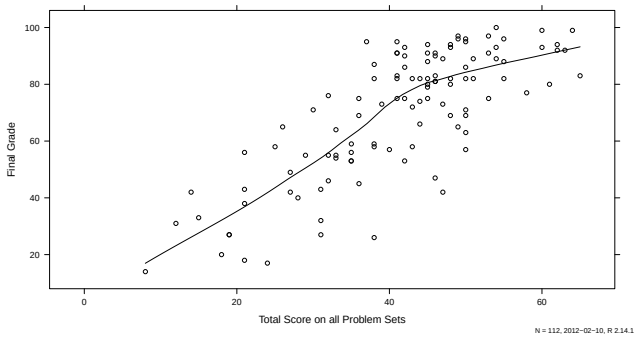



Figure 1: Problem Set Scores vs Final Grade

The increased correlation coefficient for the latter weeks of the course suggests that participants became more realistic about their own programming abilities as the semester progressed, and that in the latter half of a semester, participants may be a good source of estimations of their own success with the course.

### *3.2.2 Independent Course Work Habits*

Learners with strong independent work habits are often more successful. Since assignments were the largest 'problems' given as independent homework in this course, the learner's interactions with assignment related information was examined to better understand the relationship between independent coursework interactions and overall success. As can be seen in Figure 2, there is a relationship between the total number of times assignments were viewed and the participant's final grade (a positive correlation coefficient of .42). Even though the correlation is moderate, the relationship between final grade and assignment reading habits bears more investigation.

In order to further investigate work habits, two additional pieces of data about student work habits were calculated from the LMS logs: the number of times participants used the course LMS (views), and the number of individual days that participants used the course LMS (days active). Days active is a count of the number 24 hour periods that the participant logged in to the site at least once. Views is a count every logged interaction that a participant had with the LMS.

The number of views of LMS material has a clear relationship with the final grade of the participant (see Figure 3)

Table 2: Mean Confidence Correlations with Final Grade

| Weeks | Final Grade Correlation |
|---|---|
| 0 through 4 | .34 |
| 5 through 10 | .60 |
| Whole Semester | .52 |

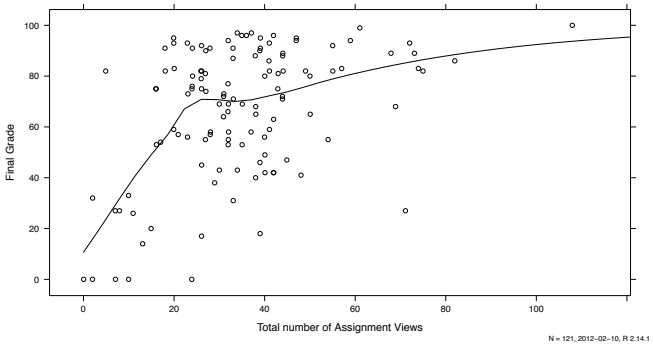



Figure 2: Total Assignment Views vs Final Grade

While the relationship is positive, a wide variance in grades is evident, especially around the 500 views point in the graph, indicating that a simple count of number of interactions is unlikely to discriminate between successful and unsuccessful learners. However, the total number of LMS views has a positive correlation coefficient of .56 with final grade.

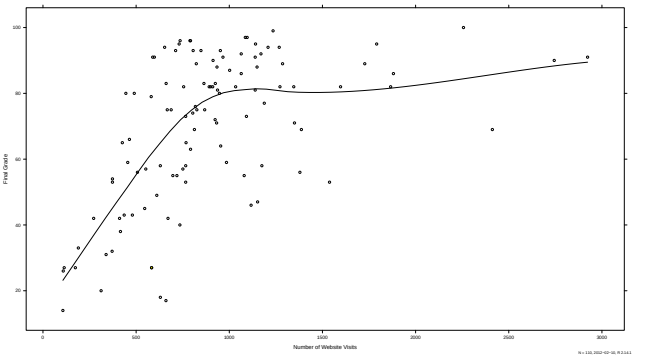



Figure 3: Total Course LMS Views vs Final Grade

The relationship between the number of days a participant was active on the LMS and their final grade is even stronger. As can be seen in Figure 4, the relationship appears to be nearly linear and the variation is similar for the entire graph. The correlation coefficient between the total number of days active and the participants' final grade is .73. It is as strong a relationship as the problem set scores, but the data capture requirements are completely passive, while problem set scores require direct action from the learner and the ability to automatically grade the practise problems.

A deeper analysis of days active provides even more fodder for consideration. When a cumulative total for days active is calculated at quarterly intervals (i.e. weeks 3, 6, 9, and 12) and a correlation coefficient is calculated for each subtotal, we find that after only three weeks of the course, the number of days active has a correlation coefficient of .53 with

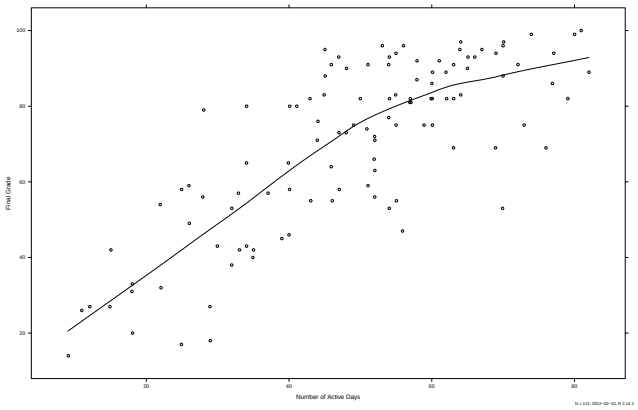



Figure 4: Total Days Active vs Final Grade

participant final grade. After six weeks of class the correlation coefficient is .62 and after nine weeks of class it is .70. Students who are active in the LMS across a number of days appear to be more successful with the course material.

## 3.3 Predicting Success

To further investigate the role of participant activity in predicting student success, decision trees were constructed using some of the study data. The trees were constructed with Rattle, a data-mining addon for the stats package R. This tree construction used the rpart(recursive partioning) model builder with the default parameters.

Table 3 shows the list of data available to the model builder.

Table 3: Data Available to Decision Tree Model Builder

| |
|---|
| final grade categories |
| participant mean confidence for the overall semester |
| participant mean confidence for weeks 0-4 and 5-10 |
| participant mean expertise for the semester |
| total LMS visits |
| total days active |
| cumulative days active for weeks 0-3, 0-6 and 0-9 |
| mean time taken to complete problem sets |

For this particular set of students, there were far more final grades of A and F than of B, C or D. In particular, very few participants received a grade of C. As a result, the grade categories of C and D were clumped together prior to constructing the decision tree and the first decision tree (Figure 5) was constructed for final grade categories of A, B, C&D, F. The discriminating attributes for the decision tree turned out to be the total number of days the participant was active on the LMS for the course, the average time a participant took to complete the weekly problem sets, and the participant's own confidence in his/her programming ability (in one case averaged over the entire semester, in others averaged over weeks 5 through 10). Two of these discriminators (days active and confidence) had high correlations with final grade, but the time taken to solve the weekly problem sets showed no correlation with final grade when examined in our preliminary work.

The decision tree shows that with passively and semi-passively

collected data, and with no direct measurement of participant domain knowledge, the extremes of success (the A learners and the F learners) can be identified with high confidence. Of particular note is the prediction of a final grade of F simply from the number of days active and the participant's own confidence in their abilities at the end of the semester. Also interesting is the predictability of a final grade of A for a confident participant who is active on the LMS for the course for more than 55 days. A high number of active days alone is enough to predict student success.

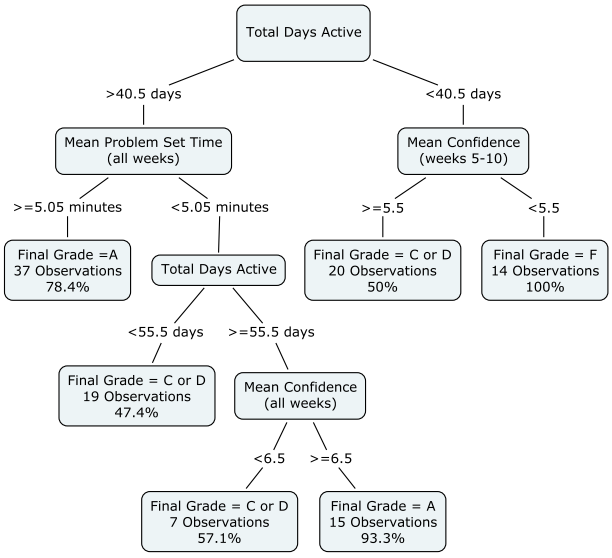



Figure 5: Decision Tree for Final Grade with D and C Clumped

A slightly different decision tree results if the aggregate bins for final grade are changed so that B and C grades are considered together, but the discriminating attributes remain the same. The second decision tree, built with the same algorithm and parameters is shown in Figure 6.

This tree predicts that if a participant used the course LMS for fewer than 40.5 days, the participant is most likely to get a D or an F as a final grade, and that the difference between a prediction of D and F is the participants own confidence in their programming. B or C grades are discriminated from A grades by the average time taken to complete the weekly problem sets (participants who spent more than 5 minutes on average were more likely to get an A).

While this analysis is still preliminary and further work will hopefully increase the confidence in the predictions, the capability to predict the extremes of success using data that can be automatically collected while students are working within a course is quite exciting.

## 4. MOVING FORWARD

This work has shown that it is possible to predict learner success without requiring data about student mastery of content (i.e. grades).

Two pieces of data stand out as a result of this investigation, the total number of days a learner is active in the course, and the scores on self-check problem sets. While both of these

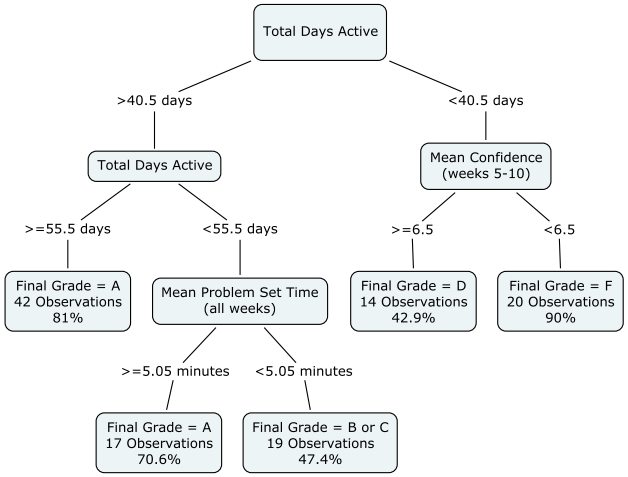



Figure 6: Decision Tree for Final Grade with B and C Clumped

pieces of data can be captured automatically, given suitable self-check questions, the number of days active meets our secondary objective of finding relevant measures that do not require examination of the learner's domain knowledge and, as such, will be the subject of our immediate detailed investigations.

This work is part of a larger effort to enhance existing Learning Management Systems with the ability to react intelligently to learner behaviours. For example, a LMS that could use its own log files to identify learners who appeared to be struggling with a courses would be extremely valuable to instructors. It could notify course instructors about students who might need extra help, and it could offer help to students. Because the underlying models in this work are based on learner behaviour rather than on evaluations of domain knowledge, such an enhanced LMS would require no subject matter knowledge and would be reusable across subject domains. The potential for such a system to improve student retention and success in both distance education and in LMS-supported face to face courses might be quite high. We are optimistic about our future endeavors.

## 5. REFERENCES

[1] S. Kotsiantis. Use of machine learning techniques for educational proposes: a decision support system for forecasting students' grades. *Artificial Intelligence Review*, pages 1–14, 2011.

[2] G. Weber and P. Brusilovsky. ELM-ART: an adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education*, 12:351—384, 2001.

# Interaction Networks: Generating High Level Hints Based on Network Community Clustering

Michael Eagle, Matthew Johnson, and Tiffany Barnes
The University of North Carolina at Charlotte
College of Computing and Informatics
9201 University City Blvd, Charlotte, NC 28223
{mjeagle, matjohns, tiffany.barnes}@uncc.edu

## ABSTRACT

We introduce a novel data structure, the Interaction Network, for representing interaction-data from open problem solving environment tutors. We show how using network community detecting techniques are used to identify subgoals in problems in a logic tutor. We then use those community structures to generate high level hints between subgoals. The preliminary results show that using network analysis techniques are promising for exploring and understanding user data from open problem solving environments.

## 1. INTRODUCTION

This paper introduces a data structure for the analysis of interaction-data collected from open problem solving environments. This structure embeds meaningful information into a complex network, which is subject to analysis through network science techniques. We apply a method of network community clustering to derive subgoals in problems. These clusters allow us to derive high-level hints, which direct students to subgoals in the open problems.

Other methods for modeling students, such as the Bayesian knowledge tracing model [5] are difficult to apply to ill defined domains, such as open problem solving tutors. First, Bayesian knowledge tracing requires each interaction to be labeled as correct or incorrect. Second, each interaction needs to be assigned a single knowledge component. For open procedural problems, both of these assumptions are challenging. As each interaction represents a step towards a goal, it is difficult to address the correctness of an individual step. While errors in the application of actions can be easily marked, errors in obtaining the correct solution require special attention. The open nature of the environment makes it possible for each interaction to provide opportunities to apply several skills. Furthermore, the skills needed for an interaction include action-application, action-opportunity recognition, and problem-solving skill.

For these reasons, many of the traditional methods for data mining on problem solving environment tutor data is either non-applicable or difficult to structure in appropriate ways. We propose using an Interaction Network as a method of structuring tutor data from problem solving environments and show one example of how this has been useful.

Next, we use the interaction network in a method for automatically generating high level hints. This is an extension of previous work where automated feedback was generated from student data [2]. We extend that work by using network community clustering on a interaction network derived from student data; this allows us to generate higher level hints based on derived subgoals. Automatically generated hints have shown positive educational results [10], and have been applied across domains [8].

## 2. PROBLEM SOLVING ENVIRONMENTS

The effect that the tutor has on how students solve problems is important. While the pedagogical benefits of scaffolded problems are well known, open problem solving environment based tutors may encourage learning in higher 'levels' of cognitive domains [4]. For this work, we define problem solving environments as non-scaffolded tutors, where students are free to apply one of many different actions and are required to complete many steps to solve a single problem, as in the Deep Thought tutor [6].

One advantage to these types of environments, compared to scaffolded problems, is that less time is required for the authoring of problems, as scaffolding is less necessary. Also there are several existing simulations and educational environments that were developed without intelligent feedback. Our goal is to provide data-driven techniques to automatically generate intelligent feedback based on previously recorded data from such environments.

## 3. INTERACTION NETWORK

In sequential problem solving environments a solution path describes a sequence of state changes from a starting position towards a desired end position. For this work we will only consider discrete time environments with deterministic state transitions. An interaction network is a data structure designed to concisely describe the information contained in a large number of such sequences. This structure is modeled as a complex weighted network, in which information relevant to educational data mining is encoded into the edges and vertices. Interaction networks provide a structure on which to perform data mining, and are also useful for visually displaying information via state diagram visualizations. The interaction network, in terms of ACT-R, is primarily concerned with the results of the Manual Control module [1]. We mention this to make a distinction between the Imaginal module, which contains steps the user makes in internal cognition. That is, the interactions are empirical observations between the subject and the tutor and do not represent internal cognitive states which may occur between

recorded actions.
An interaction network is based on individual student-tutor interactions, as recorded in the log file of the tutoring environment. We define an interaction, I, is a 5-tuple $I = (S_t, A_., S_{t+1}, U, I)$, where

- $S_t$ is the state at step t
- $A_s$ is the action performed on $S_t$
- $S_{t+1}$ is the resulting state after A has been performed
- $U$ is the unique case ID responsible for this interaction
- I is a set of additional information about the interaction. For example, $I_{time}$ would return a value for how long this interaction took. Included here are $I_{error}$, which stores the error value, and $I_{goal}$, which is true if this action resulted in a goal state.

A case represents an individual user of the tutoring system, specifically a case is a ordered pair $c = (U, I)$, where

- U is a unique identifier
- I is a set of additional information about the individual. For example, $I_{pretest}$ would return a value for this case's pretest score.

Finally, we define the interaction network for a problem $P$ is as,
$IN_P = (C, S, A, t, s, S_0, G, I_A, M)$, where

- $C$ is a set of *cases*.
- $S$ is the set of observed tutor program states
- $A$ is the set of observed actions, which connect two states
- $s : A \to S$ and $t : A \to S$ are two maps indicating the *source* and *target* states of an action
- $S_0$ is the starting state of the problem
- $G$ is the set of goal states
- $I_A : A \to I$ is a map to the source set of Interactions
- $M$ is the set of maps, which allow the lookup of relevant state, action, and case information. For example: $M_{Freq} : S \to Frequency$ will map from the state x to the frequency value for that state.

We model a solution attempt as a ordered set of interactions. We use *case* to refer to individual students, as well as student specific information. We create the interaction network for a problem by conjoining the set of all the path graphs. We use *state* to describe the state of the software environment, representing enough information so the program's state could be regenerated in the interface. We use *actions* to describe user interactions and their relevant parameters. We also store the set of all cases who visited any particular state-vertex or action-edge, allowing us to count frequencies and connect case specific information to the interaction network representation.
This representation results in a sparse, weighted, directed, labeled pseudograph, which can contain loops and cycles; with states as vertices, directed action edges to connect the states, and cases that provide additional information about states and edges. This representation allows us to build a interaction network model from any system that logs interactions in state, action, resulting-state tuples. This results in a network graph which represents the interactions of a large number of users in a relatively concise space.

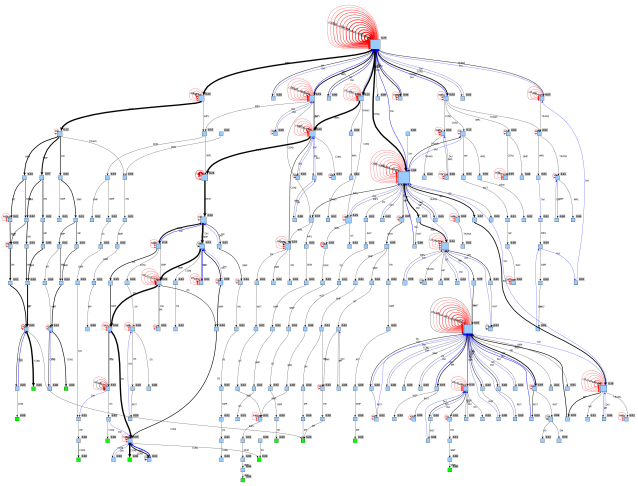


Figure 1: An Interaction Network from the Deep Thought data set. Error actions are shown as red edges, edge width depicts frequency, green squares are goal states.

## 4. THE DEEP THOUGHT TUTOR

We apply the interaction network to data from Deep Thought, a propositional logic tutor in which students are tasked with performing first-order logic proofs [6]. Students are given a set of premises and a desired conclusion; the student must then use the basic logic axioms to prove the conclusion. As the student works through the proof, the tutor records each interaction. We model the application of axioms as the actions. We model the state of the logic tutor as the conjoined set of each premise and derived proposition.
For example a student starts at state $A \vee D, A \to (B \wedge C), \neg D \wedge E$, where each premise is separated by a comma. The student performs the interaction $SIMP(\neg D \wedge E)$, applying the simplification rule of logic to the premise $\neg D \wedge E$ and derives $\neg D$. This leads to the resulting-state of $A \vee D, A \to (B \wedge C), \neg D \wedge E, \neg D$. Errors are actions performed by students that are illegal operations of logic and the tutor. For example: The student is in state $A \vee D, A \to (B \wedge C), \neg D \wedge E, \neg D$. The student performs the interaction $SIMP(A \vee D)$ in an attempt to derive $A$. The resulting-state would remain $A \vee D, A \to (B \wedge C), \neg D \wedge E, \neg D$, the log-file would mark this edge as an error.

### 4.1 Working Backwards

Deep Thought allows students to work both forward and backwards in the logic domain to solve problems [7]. Working backwards allows a student to select the goal premise and use actions to change the conclusion by adding unjustified propositions. Since students can solve a problem completely by only a single direction or a hybrid approach, the size of our state space is much larger than if only a single direction were possible, however this provides opportunities for

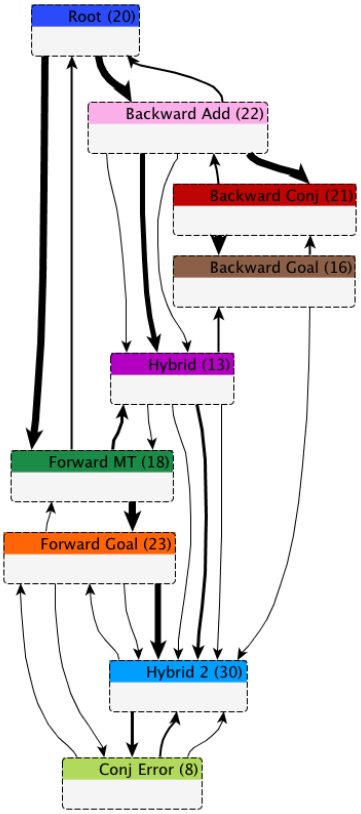



Figure 2: The Interaciton Network from figure 3 represented as a cluster graph. This network is used for the high level hinting process.

diverse problem solving techniques.

## 4.2 Data

We have six sections of student data from the year 2009, between three professors. Students were required to solve 13 problems from in the Deep Thought tutor as part of the coursework in a introduction to logic class. The problems were generally solved in order, but students could access any problem at any time. In total we have data for 303 students who submitted 1454 attempts across 13 problems totaling 64677 interactions.

# 5. GENERATING HIGH LEVEL HINTS USING NETWORK CLUSTERING

By formulating our data into an interaction network and using network invariants and metrics, we theorized we could identify sub-goals and in turn student strategies. We also theorized that problems would have underlying structures of sub-goals. We used the Girvan-Newman algorithm to cluster our interaction networks [9]. This algorithm is used to detect communities in networks, where a community has dense node-node connectedness and the edges to nodes in other communities are sparse. Since our data represents interactions, states which are similar are clustered into the same community. Performing similar actions will result in similar states. Performing similar actions in varying orders results in states which are more connected. This causes

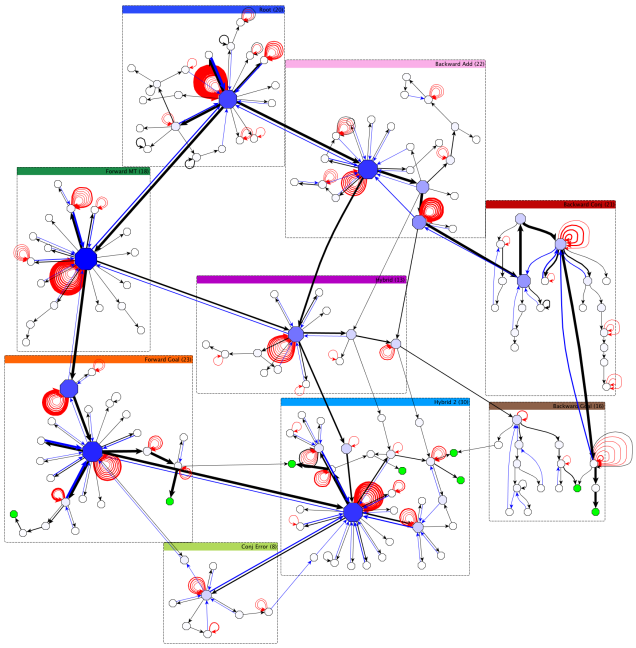


Figure 3: An Interaction Network from Deep Thought with clustering applied. This interaction network has nine clusters. Green nodes are goal states, errors are denoted by red looping edges. Blue edges are deletions where students returned to their previous state. Node size and color is based on node betweenness, and used for visual clarity.

similar interaction orders to be highly connected and result in the same community. For each sub goal, sequences of actions to reach those subgoals will be similar. Different strategies will separate different subgoals into distinct subpopulations, ie. communities. Since actions are defined by edges and Girvan-Newman separates communities by edge-betweenness, in our interaction network we are separating communities based on dominating actions.

## 5.1 Cluster Graph

In order to generate high level hints, we first divide the network into community clusters using the Girvan-Newman algorithm. This allows us to create a cluster graph, where the clusters are communities of states, highlighting important nodes we will use as sub-goals, see figure 3. The clusters can be interpreted as problem solving strategies. We manually annotated each cluster based on the strategies we observed the students using, see figure 2. We found the general strategies of working forward and backwards, as well as hybrid approaches.

To determine which hint a student receives we perform the Bellman backup algorithm to assign values to the clusters, similar to previous works by Stamper et al. [3]. Next, when a student requests a hint, we first offer a high level hint directing them to a different cluster, which represents the student's next sub-goal.

We treat each cluster as a node, with edges connecting clusters, with the same weights as in the interaction network. Each goal state is connected to a virtual exit-edge outside of it's original cluster. We assign positive values to the virtual exit-edges. Other edges are assigned a cost, to incentive

shorter solution paths. We then perform Bellman backup to generate cluster values, as is done by Stamper et al. [3] with the key distinction of using the clusters rather than using individual states. The Bellmen backup algorithm will iterate until the cluster values converge, and we use these values to provide our next cluster policy.

## 5.2 High Level Hints

By generating high level hints we can provide students with hints towards sub-goals in the problem, suggesting different strategies. We add two additional hints to Stamper's hint template, where the first two hints are based on strategies. The first hint directs students to the connected node of the 'next' cluster, if there are multiple out-degree edges from the current cluster, we can offer multiple 'next' clusters, so the student can progress along their desired strategy. Since strategies are at a higher level then individual steps, offering multiple strategies is reasonable, whereas offering multiple next steps is likely less beneficial. Students can request a second hint; this hint is based off of the parameters required to derive the first hint.

After that, if a student is unable to consider their high level strategy, we provide local hints. These hints provide immediate next steps towards the desired exit node of the current cluster. This is done by setting the exit edges of the current cluster with positive weight and performing Bellman backup. These hints are provided using the method described in Stamper [3], however it is performed locally in the current cluster, with the edges leading out of the cluster receiving the highest reward values.

All previously observed states are assigned to some cluster. If a new state is observed we can assign hints based on that student's last known cluster. This means we can still offer hints level 1 and 2 to students in states that we have not previously observed. Should students request lower level hints we can recommend students to go back to the entry point of their previous cluster. In Deep Thought students can do this by deleting prepositions from their solution through a delete action. While the interaction Network, clustering procedure, and the hint policy are domain independent, the hint templates are specific to the tutor and domain.

# 6. CONCLUSIONS

This work makes three contributions. The first is the Interaction Network, a novel data structure for modeling interaction-data from open problem solving environments. By placing interaction data into a network and encoding meaning in the network relationships, we can derive educational insight from network analysis techniques. The second contribution is the application of community detection, detecting sub-populations of the state space, to derive sub-goals in which the nodes and actions are similar. The final contribution is the method of using the community structure in the interaction network to generate both high and low level hints.

# 7. FUTURE WORK

The current algorithm used for detecting community structure, via edge betweenness, does not directly take into account problem solving specific characteristics. For example, in the Deep Thought data we have defined start and goal states. By incorporating those characteristics into the clustering algorithm more meaningful clusters may emerge.

# 8. REFERENCES


[1] J. R. Anderson. *How can the human mind occur in the physical universe?* Oxford University Press, New York, NY, USA, 2007.

[2] T. Barnes and J. Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. In *Proceedings of the 9th international conference on Intelligent Tutoring Systems*, ITS '08, pages 373–382, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] T. Barnes and J. Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS 2008)*, pages 373–382, 2008.

[4] B. S. Bloom. *Taxonomy of Educational Objectives: The Classification of Educational Goals.* Taxonomy of educational objectives: the classification of educational goals. Longman Group, New York, 1956.

[5] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278, 1994.

[6] M. J. Croy. Graphic interface design and deductive proof construction. *J. Comput. Math. Sci. Teach.*, 18:371–385, December 1999.

[7] M. J. Croy. Problem solving, working backwards, and graphic proof representation. *Teaching Philosophy*, 23:169–188, 2000.

[8] D. Fossati, B. Di Eugenio, S. Ohlsson, C. Brown, L. Chen, and D. Cosejo. I learn from you, you learn from me: How to make ilist learn from students. In *Proceedings of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*, pages 491–498, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press.

[9] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[10] J. C. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. In *Proceedings of the 15th international conference on Artificial intelligence in education*, AIED'11, pages 345–352, Berlin, Heidelberg, 2011. Springer-Verlag.

# Interleaved Practice with Multiple Representations: Analyses with Knowledge Tracing Based Techniques

Martina A. Rau
Human-Computer Interaction Institute
Carnegie Mellon University
marau@cs.cmu.edu

Zachary A. Pardos
Department of Computer Science
Worcester Polytechnic Institute
zpardos@wpi.edu

## ABSTRACT

The goal of this paper is to use Knowledge Tracing to augment the results obtained from an experiment that investigated the effects of practice schedules using an intelligent tutoring system for fractions. Specifically, this experiment compared different practice schedules of multiple representations of fractions: representations were presented to students either in an interleaved or in a blocked fashion. The results obtained from posttests demonstrate an advantage of interleaving representations. Using methods derived from Knowledge Tracing, we investigate whether we can replicate the contextual interference effect, an effect commonly found when investigating practice schedules of different task types. Different Knowledge Tracing models were adapted and compared. A model that included practice schedules as a predictor of students' learning was most successful. A comparison of learning rate estimates between conditions shows that even during the acquisition phase, students working with interleaved representations demonstrate higher learning rates. This finding stands in contrast to the commonly found contextual interference effect when interleaving task types. We reflect on the practical and theoretical implications of these findings.

## Keywords

Knowledge tracing, intelligent tutoring system, practice schedules, multiple representations, contextual interference.

## 1. INTRODUCTION

Educational data is highly complex, not only because learning is a complex process, but also because educational materials are complex. Learning materials in realistic educational settings generally cover a number of educational topics and use multiple representations. There is a substantial amount of evidence demonstrating that the use of multiple representations has a significant impact on students' learning [2,12]. When designing educational software that uses multiple representations, designers must decide how to temporally sequence the representations. In particular, it may matter whether representations are presented in a "blocked" manner (e.g., A – A – B – B) or in an interleaved manner (e.g., A – B – A – B). Research on contextual interference shows that interleaving task types leads to better learning results than blocking task types [7]. When working with multiple representations, a relevant question is whether practice with the different representations should be blocked or interleaved.

In the present paper we use log data obtained from an *in vivo* experiment (i.e., a rigorously controlled experiment in a real educational setting) that uses a successful type of intelligent tutoring system to help students learn about fractions while varying the practice schedule of multiple graphical representations. The experiment investigated the effect of practice schedules of graphical representations on students' knowledge of fractions assessed by posttests after they worked with the tutoring system. The goal of the present paper is to augment the findings from the traditional analysis of posttest data by applying a Knowledge Tracing algorithm to the log data. Analyzing student performance during the acquisition phase is particularly interesting when investigating the effects of practice schedules: a common finding is that interleaved practice schedules lead to better long-term retention and to better transfer than blocked schedules, but they often lead to worse performance during the acquisition phase [7]. Knowledge tracing, which tracks student knowledge over time, can be used to investigate learning differences between conditions during the acquisition phase [9].

In order to analyze the effect of practice schedules of multiple graphical representations on students' performance during the acquisition phase, we use a Bayesian Network model based on Knowledge Tracing [5]. Knowledge Tracing uses a two state Hidden Markov Model assumption of learning which uses correct and incorrect responses in students' problem-solving attempts to infer the probability of a student knowing the skill underlying the problem-solving step at hand. Previous research has demonstrated that extensions of knowledge tracing can be used to analyze effects of experimental conditions [9]. We combined this model with several other extensions to Knowledge Tracing to each of the four experimental conditions of the experimental study to investigate differences in model learning rates between the conditions in the Fractions Tutor.

The findings from the present paper are applicable to many other settings. Multiple graphical representations are used in a large variety of domains including science and mathematics. Whether to block or interleave representations is an important practical question in all of these domains.

## 2. THE FRACTIONS TUTORING SYSTEM

The Fractions Tutor used in the experiment was a type of Cognitive Tutor. Cognitive Tutors are grounded in cognitive theory and artificial intelligence. Cognitive Tutors have been shown to lead to substantial learning gains in a number of studies [6]. We created the tutors used in the present experiment with the Cognitive Tutor Authoring Tools (CTAT [1]. The design of the interfaces and of the interactions students engage in during problem-solving are based on a number of small-scale user studies that we conducted in our laboratory, as well as on Cognitive Task Analysis of the learning domain [3].

The Fractions Tutor included three interactive graphical representations of fractions (circles, rectangles, and number lines) and covered a comprehensive set of task types ranging from identifying fractions from graphical representations, creating graphical representations, reconstructing the unit of unit fractions and of proper fractions, identifying improper fractions from graphical representations, and creating graphical representations of improper fractions. Students solved each problem by interacting both with fractions symbols and with the interactive graphical representations. As is common with Cognitive Tutors, students received error feedback and hints on all steps. In

addition, each problem included conceptually oriented prompts to help students relate the graphical representations to the symbolic notation of fractions. These prompts were shown to be effective in an earlier study with the Fractions Tutor [12].

# 3. EXPERIMENT AND DATA

| T | Blocked | Moderate | Interleaved | Increased |
|---|---|---|---|---|
| 1 | c-c-c-c-c-c | c-c-c-r-r-r | c-r-n-c-r-n | c-c-c-c-c-c |
| 2 | c-c-c-c-c-c | r-r-r-n-n-n | c-r-n-c-r-n | c-c-c-c-c-c |
| 3 | c-c-c-c-c-c | c-c-c-r-r-r | c-r-n-c-r-n | r-r-r-r-r-r |
| 4 | c-c-c-c-c-c | r-r-r-n-n-n | c-r-n-c-r-n | r-r-r-r-r-r |
| 5 | c-c-c-c-c-c | n-n-n-c-c-c | c-r-n-c-r-n | n-n-n-n-n-n |
| 6 | c-c-c-c-c-c | c-c-c-r-r-r | c-r-n-c-r-n | n-n-n-n-n-n |
| 1 | r-r-r-r-r-r | r-r-r-n-n-n | c-r-n-c-r-n | r-r-r-n-n-n |
| 2 | r-r-r-r-r-r | n-n-n-c-c-c | c-r-n-c-r-n | n-n-n-c-c-c |
| … | … | … | … | … |
| 1 | n-n-n-n-n-n | n-n-n-c-c-c | c-r-n-c-r-n | c-r-n-c-r-n |
| 2 | n-n-n-n-n-n | c-c-c-r-r-r | c-r-n-c-r-n | c-r-n-c-r-n |
| … | … | … | … | … |

**Table I.** Practice schedule for each condition for all six task types (T). Each T was revisited three times. Students worked on nine problems per T. Each letter stands for one tutor problem and its representation: circle (c), rectangle (r), or number line (n).

The data used in this paper is based on an experimental study conducted with the Fractions Tutor during the end of the school year of 2009/2010. A total of 527 4th- and 5th-grade students from six different schools (31 classes) in the Pittsburgh area participated in the study during their regular mathematics instruction. We excluded students who missed at least one test day, and who completed less than 67% of all tutor problems (to ensure that students in the blocked condition encountered all three representations). This results in a total of $N$ = 230 (n = 63 in blocked, n = 53 in moderate, n = 52 in fully interleaved, n = 62 in increased). Students worked with the Fractions Tutor for about 5h.

Table I illustrates the practice schedules of task types and representations for the experimental conditions. In all conditions, students worked on the same sequence of task types and revisited each task type three times. Students were randomly assigned to one of four conditions. In the blocked condition, students switched between the graphical representations after 36 problems. In the moderate condition, students switched representations after every three or six problems. In the fully interleaved condition, students switched representations after each problem. In the increased condition, the length of the blocks was gradually reduced from twelve problems at the beginning to a single problem at the end. To account for possible effects of the order of graphical representations, the order in which students encountered graphical representations was also randomized.

For the experiment, students' knowledge of fractions was assessed at three test times: before their work with the Fractions Tutor, immediately after, and one week after students finished working with the Fractions Tutor. The tests included four knowledge types: area model problems (i.e., problems that involved circles and rectangles), number line problems, conceptual transfer, and procedural transfer. The results from the test data (described in more detail by [11]) showed that the fully interleaved condition performed significantly better than the blocked condition, the moderately interleaved, and the increasingly interleaved conditions on conceptual transfer at the delayed posttest. Furthermore, there was a marginally significant advantage for the increasingly interleaved condition compared to the blocked, moderately interleaved, and fully interleaved conditions on number line items at both the immediate and the delayed posttests.

The analyses presented in the current paper are based on the tutor log data obtained from the Fractions Tutor. The log data provide the number of correct steps at a student's first attempt at solving a step in the tutor, the number of attempts until a step was correctly solved, the number of hints requested per step, and the time students spent per attempt.

# 4. BAYESIAN MODEL

We evaluated four Bayesian models based on the experiment log data. Two of the models were created for the purpose of analyzing the learning rates of the conditions in the experiment while the other two were used as baseline models to gauge the relative predictive performance of the new models.

## 4.1 Learning Analysis Models

One of the simplifying assumptions made by the standard Bayesian Knowledge Tracing model [5] is that there is a probability that a student will transition from the unlearned to the learned knowledge state at each opportunity regardless of the particular problem just encountered or practice schedule of the student. Our model hypothesis corresponds to the hypothesis of the experiment that different practice schedules within a task type may be more or less effective at allowing students to acquire the skill being practiced. Thus, we depart from the Knowledge Tracing assumption of a single learning rate per skill and instead fit a separate learning rate for each of the four different practice schedule conditions defined in the experiment.

To model different learning rates within Knowledge Tracing, we adapted modeling techniques from prior work which evaluated the learning value of different forms of tutoring in (non-experiment) log data of an intelligent tutor [9]. Different representations of fractions are expected to result in different degrees of difficulty in solving the tutor problem [4]. In our condition and representation analysis model we used techniques from KT-IDEM [10] to model different guess and slips for problems depending on the representation used in the tutor problem.

We employed two models which served as benchmarks for model fit and designed two novel models for evaluating learning differences among the experiment conditions. We compared four Bayesian models all of which were based around Knowledge Tracing. Figure 1 provides an overview of the different models that we compared. The Prior-Per-Student Model [8] includes the students' individualized prior knowledge, the Condition-Analysis Model includes students' prior knowledge and models the effect of experimental condition (C). Finally, the Condition-Representation-Analysis Model incorporates students' prior knowledge (S), condition (C), and the graphical representation encountered by each student in each problem (R).

## 4.2 Model Fitting Procedure

In order to determine model fit by task type, we analyzed the log data by tasks type. For the evaluation of predictive performance, reported in the next section, a 5-fold cross-validation at the student level was used. For the reporting of learning rates by practice schedule, all data was used to train the model.

The parameters in all four models were fit using the Expectation Maximization algorithm implemented in Kevin Murphy's Bayes Net Toolbox. For the Condition-Representation-Analysis Model the number of parameters fit per task was 12 (2 prior + 4 learn rate + 3 guess + 3 slip). Probabilities of knowledge are fixed at 1 if the skill was already known, $\boldsymbol{P}(\boldsymbol{L_{n-1}})$ = 1, to represent a zero chance of forgetting, an assumption made in standard KT.

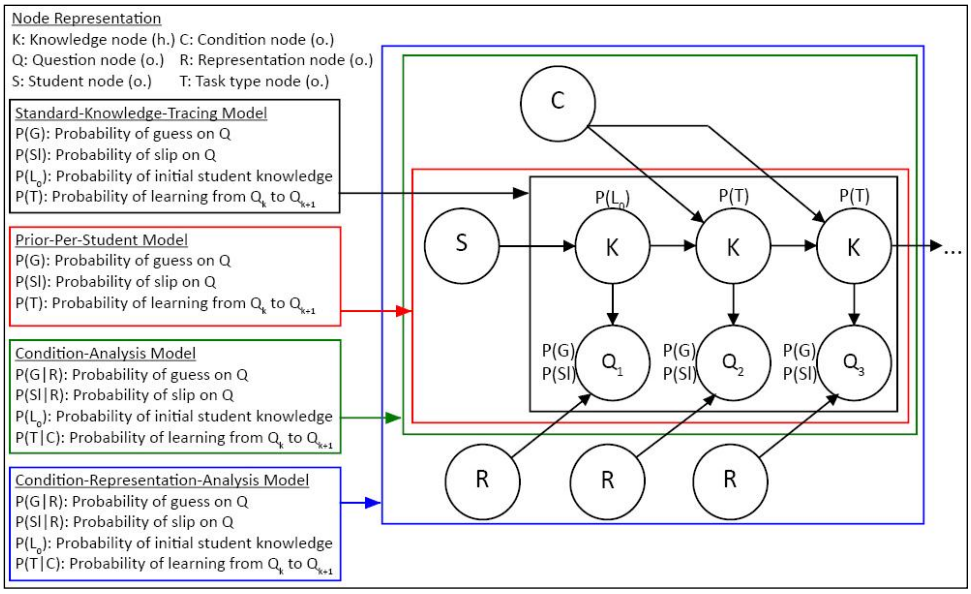



**Fig. 1.** Overview of the four different Bayesian Networks tested, with observed (o.) and hidden (h.) nodes.

## 5. EVALUATION RESULTS

| | Model | RMSE | AUC |
|---|---|---|---|
| 1 | Condition-Representation-Analysis Model | 0.3427 | 0.6528 |
| 2 | Standard-Knowledge-Tracing Model | 0.3445 | 0.6181 |
| 3 | Condition-Analysis Model | 0.3466 | 0.5509 |
| 4 | Prior-Per-Student Model | 0.3469 | 0.5604 |

**Table II.** Cross-validated prediction results summary of the four models using RMSE and AUC metrics

To evaluate the predictive accuracy of each of the student models mentioned in section 2, we conducted a 5-fold cross-validation at the student level. By cross-validating at the student level we can have greater confidence that the resulting models and their assumptions about learning will generalize to new groups of students. The metric used to evaluate the models is root mean squared error (RMSE) and Area Under the Curve (AUC). Lower RMSE equals better prediction accuracy. For AUC, a score of 0.50 represents a model that is predicting no better than chance. An AUC of 1 is a perfect prediction.

As shown in Table II, the Condition-Representation-Analysis Model has the lowest RMSE with .3427 as well as the best AUC. We conclude that the Bayesian Network that includes students' prior knowledge (S), experimental condition (C), and representations used for a certain problem (R) provides the best model fit. All predictions were statistically significantly different from one other by a paired t-test of squared errors.

Table III provides the learning rates obtained from the Condition-Representation-Analysis Model for each condition for each of the task types that the tutoring system covered. Overall, the learning rate estimates align with the results obtained from the posttest data: the interleaved condition demonstrates higher learning rates than the blocked condition. The task types were as follows: (1) identifying fractions from representations, (2) making representations of fractions, (3) reconstructing the unit from unit fractions, and (4) reconstructing the unit from proper fractions. On task type (5) identifying improper fractions from representations and (6) making representations of improper fractions.

| TT | Blocked | Moderate | Interleaved | Increased |
|---|---|---|---|---|
| **1** | 0.0061 | 0.0061 | 0.0080 | 0.0072 |
| **2** | 0.0019 | 0.0032 | 0.0065 | 0.0036 |
| **3** | 0.0149 | 0.0059 | 0.0337 | 0.0030 |
| **4** | 0.0037 | 0.0022 | 0.0035 | 0.0014 |
| **5** | 0.0108 | 0.0220 | 0.0124 | 0.0130 |
| **6** | 0.0043 | 0.0107 | 0.0078 | 0.0090 |
| **Overall** | 0.0062 | 0.0056 | 0.0120 | 0.0062 |

**Table III.** Learning rates by task type (TT) and condition from the Condition-Representation Analysis Model.

The learning rates by task type provide more specific information on the nature of the differences between conditions in learning rates. For all but the fourth task type, the fully interleaved condition demonstrates a higher learning rate than the blocked condition. This difference was statistically significant for tasks 1, 2 and 3 ($p < 0.05$) and moderately significant for task 5 ($p = 0.06$). The same binomial test as was used in [9] was employed here to test for significance. The interleaved condition achieved the highest overall learning rate which was twice that of any other condition. This was despite having the second lowest percent correct among responses in the acquisition phase.

## 6. DISCUSSION

The findings from the Bayesian Networks support and augment the findings from the posttest data in several ways. First, the finding that the Condition-Representation Analysis Model provides the best fit to the log data confirms the overall finding from the posttest data of the experiment that practice schedules of multiple representations matter. It also highlights that per item level parameters are greatly beneficial, especially when the problem opportunities involve different cognitive operations, such as solving problems with different representations. Furthermore, the finding that the representation used in a tutor problem is a useful predictor of learning confirms that different graphical representations provide different conceptual views on fractions in a way that influences how students understand fractions [4].

Second, the learning rate estimates per condition support the finding from the posttest data that interleaved practice schedules of multiple graphical representations of fractions lead to better learning than blocked practice schedules. This finding is interesting, because the literature on contextual interference shows that interleaved practice schedules often impair performance during the acquisition phase [7]. It is assumed that temporal variation between consecutive problems interferes with immediate performance since students have to adapt their problem-solving procedures each time they encounter a new task. This interference leads to higher processing demands and lower performance during the acquisition phase, but results in better long-term retention and transfer performance later on. Hence, one might expect that higher learning gains in the interleaved condition become apparent only in the posttest data, but not during the acquisition phase, because they might be "masked" by impaired performance due to interference. Our findings show, however, that an intervention that is assumed to lead to impaired performance during the acquisition phase nonetheless leads to a learning advantage that is not only detectable in higher posttest performance but also during the acquisition phase using our experiment adapted Bayesian model. Bayesian Network analyses allowed us to detect learning gains that may be too subtle to detect during the acquisition phase when relying on performance. We believe this was able to be achieved thanks to the item level modeling that distinguished learning from variation in problem difficulty.

Finally, the differences between learning rate estimates between task types yield important insights into the effectiveness of the tutor task types that will help improve the tutoring system in future iterations. Bayesian Network analysis provides us with a useful tool that can help us evaluate this iterative improvement of the tutoring system at a much finer grain size than through the traditional analysis of posttest data. This technique also allowed for analysis to be accomplished without pre or post test data.

The results from the Bayesian Network analysis presented in this paper yield interesting insights that are both of theoretical and practical significance. Our results confirm the finding from our previous experiment [18] that interleaving representations leads to better learning than blocking representations and extend the finding by demonstrating that the advantage of interleaved practice is apparent also during the acquisition phase. This finding is of practical relevance as it demonstrates that face-value methods, such as percent correct during the acquisition phase, do not provide sufficient information to evaluate an educational intervention. Since many domains use multiple graphical representations to augment instructional materials, we believe that our findings have the potential to generalize across a wide range of learning materials. Furthermore, the analysis of learning rates by condition allows us to identify parts of the Fractions Tutor curriculum that need to be improved as they do not seem to help students learn. Bayesian Network analyses can help us make sense of the complex educational data that we obtain from the rich settings in which education takes place, and hence, help us understand complex learning processes.

## 7. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation, REESE-21851-1-1121307, "Graduates in K-12 Education" Fellowship, award number DGE0742503, and by the Pittsburgh Science of Learning Center which is funded by the National Science Foundation, award number SBE-0354420. In addition, we would like to thank Haya Shamir, Ken Koedinger, John Stamper, and the students and teachers who participated in our study.

## 8. REFERENCES

[1] Aleven, V., Mclaren, B. M., Sewall, J., & Koedinger, K. R. 2009. A new paradigm for intelligent tutoring systems: Example-tracing tutors. International Journal of Artificial Intelligence in Education, 19(2), 105-154.

[2] Ainsworth, S., Bibby, P., & Wood, D. 2002. Examining the effects of different multiple representational systems in learning primary mathematics. Journal of the Learning Sciences, 11(1), 25-61.

[3] Baker, R. S. J. D., Corbett, A. T., & Koedinger, K. R. 2007. The difficulty factors approach to the design of lessons in intelligent tutor curricula. International Journal of Artificial Intelligence in Education, 17(4), 341-369.

[4] Charalambous, C. Y., & Pitta-Pantazi, D. 2007. Drawing on a Theoretical Model to Study Students' Understandings of Fractions. Educational Studies in Mathematics, 64(3), 293-316.

[5] Corbett, A. T., & Anderson, J. R. 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, 4, 253-278.

[6] Corbett, A. T., Koedinger, K., & Hadley, W. S. 2001. Cognitive tutors: From the research classroom to all classrooms. In Technology enhanced learning: Opportunities for change. (pp. 235-263). Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers.

[7] De Croock, M. B. M., Van Merrienboer, J. J. G., & Paas, F. G. W. C. 1998. High versus low contextual interference in simulation-based training of troubleshooting skills: Effects on transfer performance and invested mental effort. Computers in Human Behavior, 14(2), 249-267.

[8] Pardos, Z. & Heffernan, N. (2010) Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence ofthe Expectation Maximization algorithm. In Baker et al. (Eds.) Proceedings of the 3rd International Conference on Educational Data Mining (pp. 161-170).

[9] Pardos, Z. A., Dailey, M. D., Heffernan, N. T. (2010) Learning what works in ITS from non-traditional randomized controlled trial data. In Aleven et al. (Eds.) Proceedings of the 10th International Conference on ITS2010 (pp. 41-50).

[10] Pardos, Z. & Heffernan, N. (2011) KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model. In Konstant et al. (Eds.) In Proceedings of the 20th International Conference on UMAP 2011 (pp. 243-254).

[11] Rau, M. A., Aleven, V., Tunc-Pekkan, Z., Pacilio, L., & Rummel, N. (accepted). How to schedule multiple graphical representations? A classroom experiment with an intelligent tutoring system for fractions. To appear in the proceedings of ICLS 2012.

[12] Rau, M. A., Aleven, V., & Rummel, N. 2009. Intelligent Tutoring Systems with Multiple Representations and Self-Explanation Prompts Support Learning of Fractions. In Dimitrova et al. (Eds.), Proceedings of the 14th International Conference on AIED (pp. 441-448).

# Learning Gains for Core Concepts in a Serious Game on Scientific Reasoning

Carol Forsyth, Philip Pavlik, Jr. Arthur C. Graesser, Zhiqiang Cai, Mae-lynn Germany
The University of Memphis
Institute for Intelligent Systems
400 Innovation Drive
Memphis,TN 38105
cmfrsyth,ppavlik,graesser,zcai mlgerman@memphis.edu

Keith Millis
Northern Illinois University
Psychology Building
Dekaulb,IL 60115
kmillis@niu.edu

Robert P. Dolan
Pearson Education
184 N Leverett Rd.
Leverett, MA 01054
bob.dolan@pearson.com

Heather Butler, Diane Halpern
Claremont Graduate University
Psychology Department
Claremont, CA 91711
hbutler, diane.halpern@cmc.edu

## ABSTRACT

*OperationARIES!* is an Intelligent Tutoring System that teaches scientific inquiry skills in a game-like atmosphere. Students complete three different training modules, each with natural language conversations, in order to acquire deep-level knowledge of 21 core concepts of research methodology (e.g., correlation does not mean causation). The student first acquires basic declarative knowledge and then applies the knowledge by critiquing case studies on scientific methodology and finally generating questions that reflect the core topics. A study using a pretest-training-posttest design was conducted in which 46 college students interacted with the modules of *OperationARIES!*, resulting in thousands of logged measures. The goal of this investigation was to discover the different trajectories of learning within 11 of these core concepts by evaluating 3 main constructs (e.g., discrimination, generation, and time on task) represented by key logged measures. Different constructs showed relationships with specific core concepts. Three core concepts were analyzed with stepwise regression and 5-fold cross-validation in order to discover contributing factors to learning gains for these core concepts.

## Keywords

Intelligent Tutoring Systems, reasoning, serious games, research methods, discourse

## 1. INTRODUCTION

Social scientists often emphasize differences among students in their analyses of learning. The present research acknowledges such differences among students and aptitude-treatment interactions [1]. However the salient message in this study puts the magnifying glass on differences between core concepts in a subject matter. Simply put, the learning trajectories of core concepts may differ substantially depending on their content, complexity, and difficulty.

### 1.1 Cognitive Constructs Predicting Learning

The cognitive and learning sciences have identified principles of learning that offer likely hypotheses regarding differences in learning trajectories for core concepts [2]. Some concepts are learned by simply spending time reading and studying the material, a factor called time on task [3]. Time on task is normally optimized when concepts are presented on multiple occasions and distributed over time rather than concentrated in one time block [2, 4, 5]. Some concepts are learned primarily by actively generating the associated information about the concepts [2,4], particularly explanations [2, 5, 6, 7, 8]. Some concepts are best learned by testing experiences [9 ] and feedback on their answers [10], whereas others are best learned by either tutorial interaction [8, 11, 12, 13], scaffolding to get the student to generate good questions about difficult conceptualizations [14, 15], or tasks to get the student to make important discriminations among alternatives [8, 11, 13, 14]. The present study investigates the training events and experiences that contribute to the acquisition of critical core concepts. Our central point is simple. Core concepts have idiosyncratic characteristics that lend themselves to particular learning activities that optimize their acquisition.

The goal of this investigation is to discover the cognitive factors that predict the learning of core concepts in research methodology. The concepts range from concrete to abstract topics [8, 11, 14] and may require the student to utilize different skills. For example, understanding the meaning of an operational definition may be quite shallow in nature and possibly only require more time on task. Conversely, a more challenging abstract topic such as

correlation vs. causation may not be mastered by simply memorizing a definition but rather by higher level reasoning, discrimination among similar constructs, and generating ideas or questions. The learning environment is a serious game called Operation ARIES, as described next [12]. Although we have considered thousands of measures collected during 20 hours of training in ARIES, our analyses converged on three broad time-honored constructs in the cognitive and learning sciences: time on task, discrimination, and generation.

## 1.2 Operation ARIES: A Serious Game

*OperationARIES!* (called ARIES for short, an acronym meaning Acquiring Research Investigative and Evaluative Skills) is an Intelligent Tutoring System that has an embedded storyline and game-like elements to engage students as they learn research methodology. The narrative includes alien invaders who have come to take over the world by presenting bad science. The student player joins forces with the Federal Bureau of Science in order to save the world from this threat. The storyline and the iterative presentation of these topics are presented to the students across three specific ARIES modules (i.e., Training module, Case Study module, and Interrogation module), each focusing on different types of knowledge acquisition: didactic knowledge, application, and question generation. The learner interacts in natural language conversations with multiple artificial agents in order to learn 21 core concepts of research methodology.

In the Training module students learn didactic knowledge by reading an E-text, answering multiple choice questions, and having dynamic tutorial conversations with two pedagogical agents about the 21 core concepts. In the Case Study module, students apply the knowledge by conversing with three artificial agents while identifying flaws in research cases with the aid of both a list of 12 potential flaws and the E-book. Finally, in the Interrogation module, students pose questions to an artificial agent in order to decide if the research case is sound. The learner is aided by a score-card which provides immediate feedback as well as suggested questions. The flaws covered in the Case Study module and Interrogation module are aligned with the core concepts in the Training module.

This paper explores the specific cognitive activities in this serious game that predict learning of a subset of the 21 core concepts. These cognitive activities are part of the Training, Case Study, and Interrogation modules.

# 2. METHODS

The participants were 46 students at 2 separate schools in Southern California. There was a pretest-training -posttest design, with two versions of a test that were counterbalanced between pretest and posttest. All of the students were enrolled in research methodology courses taught by the same instructor. The pretest and posttest consisted of open-ended and multiple-choice questions about the 21 core concepts. The participants interacted with the Training module in pairs, alternating between actively typing into the system and passively observing their human partner interacting (a difference that was not analyzed in this study). The participants intermittently answered survey questions about the storyline and tutorial conversations, but these measures are not investigated in the current study. The alternation between partners as well as the surveys did not occur in the latter two modules (Case Study and Interrogation).

## 2.1 Measures

The log files of ARIES had thousands of measures including fine-grained measures for each module. Measures include latency measures, string variables and virtually every aspect of the typed interaction. With so many variables, the focus of this particular investigation will be on those measures that funnel into the three constructs of time on task, generation, and discrimination.

Each of the 3 constructs was represented by a unique indicator for each module. Specifically, time on task was represented in the Training module by reading times per page in the E-Text, whereas the time spent on cases was the measure for the Case Study and Interrogation modules. In order to assess generation, the measures consisted of the number of words articulated by the student in conversational turns for each module. Discrimination scores were collected in each module. The Training module used the multiple-choice performance scores (0 to 1). In the Case Study module, a discrimination score was calculated by subtracting the proportion of false alarms from hits as reflected by the match scores of the language processing algorithms within the system. The Interrogation module also used signal detection components derived from student performance on the score-card that discriminated whether a flaw was or was not present in a study.

In order to measure learning gains, we computed proportion scores for the pretest and posttest. Each test consisted of a multiple-choice and short-answer question corresponding to each of the 21 concepts. Proportional learning gains scores [(posttest-pretest)/(1-pretest)] were calculated in order to adjust for the variation of prior-knowledge across the students. These scores were available for each of the 21 concepts.

# 3. ANALYSES

Although this original dataset consisted of 46 participants, 10 of the subjects were removed due to extensive amounts of missing data (i.e. usually more than one module). Of the remaining 36 students, mean values were used to replace the missing data for discrimination scores. However, time on task and generation scores were simply left as 0's.The most complete set of original data, prior to mean replacements were available for 11 core concepts.

These core concepts were presented and tested across all three modules, so they were selected in the subsequent analyses. .

## 3.1 Correlations

The proportional learning gain scores ranged from .17 (Causal Claims) to .50 (Subject Bias), with a mean of .34 over the 11 core concepts. We computed correlations between these gain scores and the training process measures. We found a number of significant correlations, but the more important conclusion is that the profile of process to learning correlations differed greatly among core concepts.

It is beyond the scope of this report to present the full set of data. Instead, we will focus on a few core concepts that illustrate the differences. For example, the Training module reigned in the learning of one core concept (Objective Scoring of the Dependent Variable) when inspecting the correlations, which were significantly positive for the three measures: reading time, words generated, and discrimination. In contrast, the Interrogation module was most important for Subject Bias, where the corresponding three measures had significant correlations.

The differences in learning process profiles among core concepts underscores our central claim that core concepts vary considerably in learning trajectories.

## 3.2 Stepwise Regressions and Cross-Validation

We performed analyses on three core concepts that had distinctive profiles of correlations. These included Objective Scoring, Subject Bias, and Causal Claims. Each of these core concepts was analyzed separately using stepwise regressions with predictor variables that included those with the highest correlations ($r$ >|.2|) with proportional learning gains. The resulting model was then cross- validated using a 5-fold procedure with 4 folds for training and 1 for test..

### *3.2.1 Objective Scoring of the Dependent Variable*

This core concept showed correlations with the proportional learning gains for the reading times (time on task measure, $r$ = .32, $p$<.05) and the multiple choice questions (discrimination score, $r$ = .32, $p$<.05) in the Training module. In all 3 modules, the number of words generated significantly correlated with proportional learning gains (Training ($r$ = .42, $p$<.05); Case Study($r$ =.28, $p$<.05); Interrogation ($r$ =.28, $p$ <.05). When these significant correlates were entered into a stepwise regression, the analysis removed the time allocated to multiple choice questions (time on task) and the words generated in the Training module, thereby converging on a model that includes words generated in the Interrogation module and the Case Study module and the reading times from the Training module ($F$ (3, 33) = 4.91, $R^2$ = .31, $p$<.05). In the full model, the words generated in the Interrogation module had a marginally significant main effect ($F$ (3, 33) = 3.61, $p$ = .06); the words generated in the Case Study module did not have a significant main effect ($F$ (3, 33) = 2.45, $p$ = .13), but reading times were significant ($F$ (3,33) = 8.67, $p$<.05). Given these results, a second model was created using the generation score for the Interrogation module and the reading times. The model was significant ($F$ (2, 34) = 4.338, $R^2$= .20, $p$<.02) with a marginally significant main effect for generated words ($F$ (2, 34) = 3.23, $p$ = .08) and a significant main effect for reading times ($F$ (2, 34) = 5.45, $p$<.05). When this model was cross validated, the training set accounted for 26% of the variance ($R^2$= .26), and a test set accounted for 25% of the variance ($R^2$ = .25)

### *3.2.2 Subject Bias*

For this core concept, the variables with the highest correlations with learning gains were the multiple choice discrimination score from the Training module ($r$ =.20, $p$<.10), and the discrimination ($r$ = .20, $p$<.1), generation ($r$ =.33, $p$<.05), and time on task ($r$ = .26, $p$<.05) measures from the Interrogation module. With all predictors entered into a stepwise regression, the resulting significant model included only the words per case (generation) and the discrimination score from the Interrogation module ($F$ (2, 34) = 3.304, $R^2$= .16, $p$<.05). Upon further examination, there is a significant main effect for generation ($F$ (2, 34) = .498, $p$<.05) but not for the discrimination score ($F$ =1.63, $p$>.05). A second linear model with just the generation score was significant model ($F$ (1, 35) = 4.368, $R^2$= .11, $p$ <.05). Next, the significant generation predictor only was cross-validated using a 5-fold cross validation procedure resulting in a training set predicting 8% of the variance ($R^2$ =.08) and a test set predicting 6% of the variance ($R^2$ = .06). However, we are still tentative about drawing strong conclusions from this because of the low power in detecting differences in the regression.

### *3.2.3 Causal Claims*

This core concept had low learning gains (.17) compared with the other topics. The two variables with highest correlations for learning were discrimination from the Case Study module ($r$ = .28, p<.05) and the generation metric in the Interrogation module ($r$ = .23, $p$ <.1). However, a follow-up analysis with stepwise multiple regression was only marginally significant ($F$ (2, 34) = 2.863, $R^2$ = .14, $p$=.07) and cross validation assessments were not significant.

## 4. CONCLUSIONS

Our analyses revealed very different learning profiles for specific core concepts in research methodology. The value of the didactic Training module was most pronounced for Objective Scoring of Dependent Variables, whereas the Interrogation module was most successful for Subject

Bias, and Case Study was most promising for Causal Claims. The constructs of time on task, generation of information, and discrimination were also quite different for the different core concepts. Moreover, students did not learn much about differentiating causal from correlational claims. This topic may be very abstract to many students, difficult to comprehend, and in need of substantially more training.

One important implication of this study is that the different core concepts might be assigned different modules or a different amount of training allocated to each module. For some core concepts, it may be sufficient to have them read text and prompt them to articulate propositions in language. For other core concepts, they need a large number of case study examples to apply their knowledge in a discriminating fashion. Simply put, training experiences need to be optimally allocated to the constraints of content.

There are a number of limitations in this study that prevent us from making more definitive claims about the type of training that should be matched to our core concepts. The study had a low number of participants and a moderate number of missing values for observations. However, we can confidently state that correlations between learning gains and the key constructs of generation, discrimination and time on task do vary across core concepts of research methodology in *OperationARIES!*.It is important to explore different learning trajectories of specific core concepts in addition to differences among students.

## 5. ACKNOWLEDGEMENTS

This research was supported by the Institute for Education Sciences (R305B070349) and National Science Foundation (HCC 0834847). The opinions expressed are those of the authors and do not represent views of the IES and NSF.

## 6. REFERENCES

[1] Cronbach, L. and Snow, R. Aptitudes and Instructional Methods: A Handbook for Research on Interactions. New York: Irvington, 1977.

[2] Pashler, H., Bain, P. M., Bottge, B. A., Graesser, A., Koedinger, K., and McDaniel, M. Organizing Instruction and Study to Improve Student Learning: IES Practice guide. (NCER 2007-2004). Washington, DC: National Center for Education Research, 2007.

[3] Taraban, R., Rynearson, K., and Stalcup, K. Time as a variable in learning on the world wide web. Behavior Research Methods, Instruments, & Computers, 33 (March 2001), 217-225.

[4] Cepeda, N. J., Pashler, H., Vul, E., Wixted, J. T. and Rohrer, D. Distributed practice in verbal recall tasks: A review and quantitative synthesis. Psychological Bulletin, 132 (September 2006), 354-380.

[5] Kopp, K., Britt, A., Millis, K., and Graesser, A. Improving the efficiency of dialogue in tutoring. Learning and Instruction, (in press).

[6] McNamara, D. S., O'Reilly, T. P., Best, R. M., and Ozuru, Y. Improving adolescent students' reading comprehension with iSTART. Journal of Educational Computing Research, 34 ( June 2006), 147-171.

[7] Roscoe, R.D. and Chi, M.T.H. Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. Review of Educational Research, 77 (December 2007), 534-574.

[8] VanLehn, K., Graesser, A.C., Jackson, G.T., Jordan, P., Olney, A., and Rose, C.P. When are tutorial dialogues more effective than reading? Cognitive Science, 31 (January 2007), 3-62.

[9] Roediger, H. L. III. and Karpicke, J. D. The power of testing memory: Basic research and implications for educational practice. Psychological. Science, 1 (September 2006), 181-210.

[10] Shute, V. J. Focus on formative feedback. Review of Educational Research, 78 (March 2008), 153-189.

[11] Graesser, A.C., Conley, M., and Olney, A. Intelligent tutoring systems. In K.R. Harris, S. Graham, and T. Urdan (eds.), APA Educational Psychology Handbook: Vol. 3. Applications to Learning and Teaching(2012). Washington, DC: American Psychological Association, 451-473.

[12] Millis, K., Forsyth, C., Butler, H., Wallace, P., Graesser, A., Halpern, D. Operation ARIES!: A serious game for teaching scientific inquiry. In M.Ma, A. Oikonomou & J. Lakhmi (eds.) Serious Games and Edutainment Applications (in press), Springer-Verlag, London, UK.

[13] Ritter, S., Anderson, J. R., Koedinger, K. R., and Corbett, A. Cognitive tutor: Applied research in mathematics education. Psychological Bulletin and Review, 14 (April 2007), 249-255.

[14] Graesser, A. C., Ozuru, Y., and Sullins, J. What is a good question? In M. G. McKeown and L. Kucan (eds.) Threads of coherence in research on the development of reading ability (2009), New York: Guilford, 112-141.

[15] Graesser, A. C., and Person, N. K.. Question asking during tutoring. American Educational Research Journal, 31 (Spring 1994),104-137.

---

## About the authors:

**Carol Forsyth** is currently a PhD student in Experimental Psychology at the University of Memphis. She is also pursuing Cognitive Science graduate certification. Her main areas of interest include discourse processes, serious games, educational data mining, and Intelligent Tutoring Systems. Forsyth works as a graduate research assistant to Dr. Arthur Graesser, a Full Professor of Psychology at the University of Memphis, Co-Director of The Institute for Intelligent Systems, Senior Research Fellow at The University of Oxford and author of over 500 journal articles. In addition to Dr.Graesser, Forsyth has been fortunate to work with and learn from all of the co-authors on this paper.

# Leveraging First Response Time into the Knowledge Tracing Model

Yutao Wang
Worcester Polytechnic Institute
100 Institute RD.
Worcester, MA, USA
yutaowang@wpi.edu

Neil T. Heffernan
Worcester Polytechnic Institute
100 Institute RD.
Worcester, MA, USA
nth@wpi.edu

## ABSTRACT

The field of educational data mining has been using the Knowledge Tracing model, which only look at the correctness of student first response, for tracking student knowledge. Recently, lots of other features are studied to extend the Knowledge Tracing model to better model student knowledge. The goal of this paper is to analyze whether or not the information of student first response time of a question can be leveraged into Knowledge Tracing model and improve Knowledge Tracing's prediction accuracy. In our experiments, we used discretized first response time data to predict students' correctness of the next question, and leveraged the result into a Knowledge Tracing model. Our analysis confirmed the value of student first response time in modeling student knowledge.

## Keywords

Educational data mining, intelligent tutoring systems, student modeling, first response time.

## 1. INTRODUCTION

Modeling student behavior is crucial for education. For decades, researchers in the field of educational data mining (EDM) have been developing various methods of modeling student behavior using their performance as observations. One example is one of the dominant student model called Knowledge Tracing (KT) model built by Corbett and Anderson in 1995[1], which uses a dynamic Bayesian network to model student learning. Recently, lots of other features are studied in the framework of the Knowledge Tracing model to extend the Knowledge Tracing model to better model student knowledge. These features include the difficulty of problems [2], if it is a new day since a student last saw a problem [3], the assistance students require in answering a problem [4], etc. This paper analyses another piece of information: student first response time. We want to find out if students' first response time of a question can be used for improve KT's prediction accuracy.

Student response time, as an important feature that characterizes student behavior, is studied in the field of Intelligent Tutoring Systems in various models either due to its subjective importance or after some data analysis.

Some of these models use response time for understanding students' behaviors during problem solving in tutoring systems. Beck J.E. 2005 [5] used response times to model student disengagement; Shih B. et al. 2008 [6] built a response time model for bottom-out hints as worked examples; Arroyo I. et al. 2010 [7] used time required to solve a problem to model student effort.

Some models use different time information as one of many features in their models to indicate student knowledge. Such as Rai and Beck 2011[8] used the average time spent on each attempt in modeling their game-like math tutor.

Those works did not focus on using student first respond time as a direct indicator of student knowledge.

### 1.1 The Tutoring System

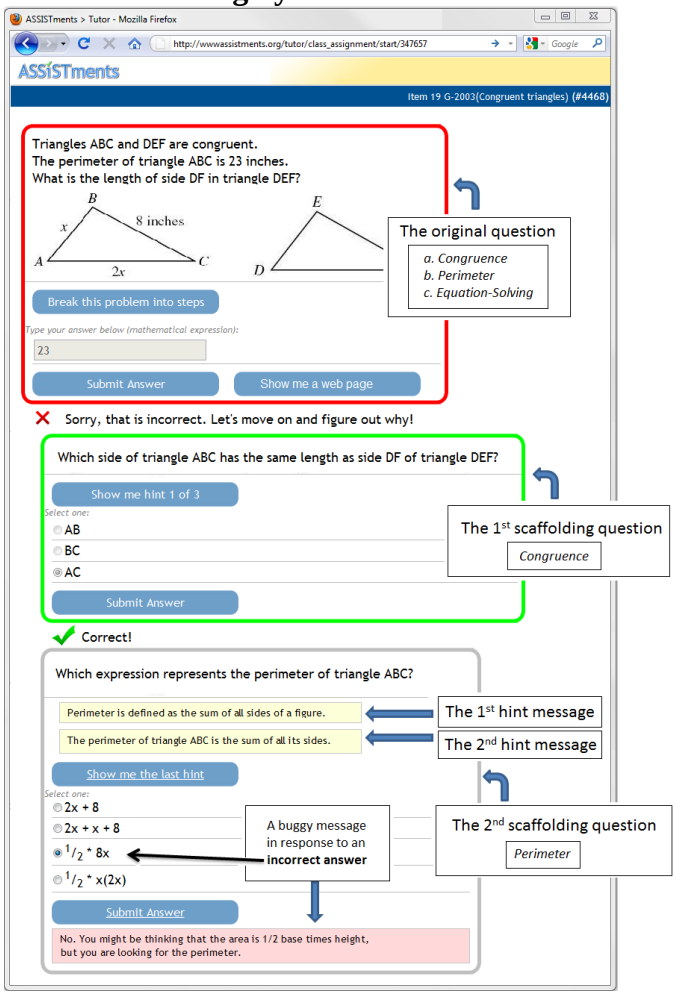



**Figure 1 A typical senerio in ASSISTments system.**

The data used in the analysis came from the ASSISTments system, a freely available web-based tutoring system for 4th

through 10th grade mathematics (approximately 9 through 16 years of age). The system is mainly used in urban school districts of the Northeast United States. Students use it in lab classes that they attend periodically, or for doing homework at night.

The system provides tutorial assistance as buggy messages or scaffolding questions if a student makes a wrong attempt, and hint messages if a student asks for help. Figure 1 shows an example scenario in the ASSISTments system.

## 1.2 The KT Model

The Knowledge Tracing model shown in Figure 2 has been widely used in ITS and many variants have been developed to improve its performance (Baker et al. 2010, Pardos and Heffernan 2010). It uses 4 parameters for each skill, with two for student knowledge and the other two for student performance. The parameters prior knowledge and learning are called learning parameters. Prior knowledge is the likelihood the student knows the skill when he/she first uses the tutor. Learning is the probability a student will acquire a skill as a result of an opportunity to practice it. The parameters slip and guess are called the performance parameters in the model. An assumption of this model is that even if a student knows a skill, there is a chance he/she might still respond incorrectly to a question of that skill. This probability is the slip parameter. Conversely, a student who does not know the skill might be able to generate a correct response. This probability is referred to as the guess parameter.

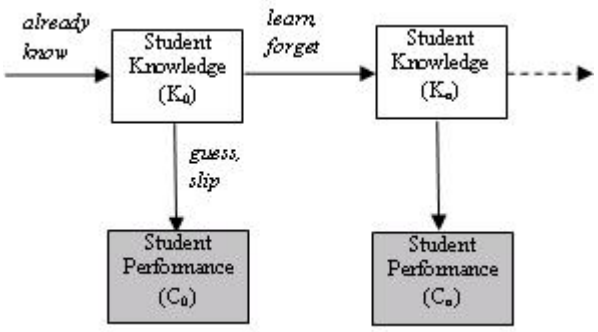



**Figure 2. Knowledge Tracing model**

*Prior Knowledge = Pr (K0=True)*
*Guess = Pr (Cn=True | Kn=False)*
*Slip = Pr (Cn=False | Kn =True)*
*Learning rate = Pr (Kn =True | Kn−1=False )*

In our experiment, we used the Bayes Net Toolbox for Matlab developed by Murphy (2001) to implement Knowledge Tracing, and the Expectation Maximization (EM) algorithm to fit the model to the dataset. The EM algorithm finds a set of parameters that maximize the likelihood of the data by iteratively running an expectation step to calculate expected likelihood given student performance data and a maximization step to compute the parameters that maximize that expected likelihood. There have been reported issues of local maxima when using the EM algorithm. Pardos and Heffernan (2001) concluded, based on a simulation study, that with the initial parameters of this algorithm in a reasonable range (the sum of initial guess and slip value is smaller than 0.5), the algorithm will always converge to a point near the true parameter value. In our experiments, we choose initial parameters for each skill as follows*: initial knowledge* = 0.5, *learning* = 0.1, *guess* = 0.1, *slip* = 0.1. These initial parameters are set to be similar with the results of previous experiments that estimated the Knowledge Tracing model parameters on some other datasets from the ASSISTments system.

# 2. PROBLEM AND APPROACH

Although there has been study done in both student response time and student knowledge, there is no research in using student response time to indicates student knowledge. In this paper, we focus on leveraging student first response time into the Knowledge Tracing model to see whether or not student first response time is valuable in modeling student knowledge and enhance KT model's prediction accuracy of student performance.

There are various explanations in different student first response time. For example, a short first response time could either mean the student is proficiency on the skill or the student is guessing the result or gaming the system; also, a long first response time could either mean the student is thinking about the given problem or he/she is just doing some off task behavior. As a result, the connection between student first response time and student knowledge could be blurred by many other factors. However, since student response time is one of the most important information of student behavior that could be easily gathered by Intelligent Tutoring Systems, analyses on its ability of modeling student knowledge and improving performance prediction is still meaningful to this field. To handle the other factors that could influence the result, we discretized the first response time data to eliminate unnecessary details of the information, and aim for finding the general indication of this information towards student knowledge and future performance.

## 2.1 Data

The data we analysed are from school year September 2010 to September 2011, which consisted of 15931 students who solved at least 20 problems within ASSISTments. We filtered out skills that have fewer than 50 students and randomly selected 2015 student users. As a result, we have 498 ,988 data records. Each data record is recorded right after a student answered a problem, and logged relevant information including the identity of the student, the problem identity and skills required to solve it, the correctness of the student's first response to this problem, the first response time the student spent on this problem, and the timestamp when the student start and finish solving this problem.

## 2.2 Discretization of First Response Time

As we discussed before, since student first response time includes information other than student knowledge. To eliminate unnecessary details of the information, which could be relevant to other factors, we discretized student first response time data into several bins.

Our goal is to find out if the main character of student first response time contains unique information about student knowledge in compare with other features. We discretized student first response time data into four categories. The way we define these categories are based on the follows assumptions.

The first assumption is, in general, students that need more time to first respond to a problem have lower knowledge than students that need less first response time, because the former require more time to answer the question.

The second assumption is, in general, the data records that show extremely little time of student first response time are likely to indicate some special behaviors such as gaming, thus, the first response time in those data records may not be as useful in indicating students knowledge.

The third assumption is, in general, the data records that show extremely long time of student first response time are also likely to indicate special behaviors such as off task behaviors, thus, the first response time in those data records may also not be useful in indicating student knowledge.

According to these assumptions, the four categories of student first response time are: extremely short, short, long, extremely long.

Also, considering student first response time highly varies by problem, we computed different cut points of these four categories for each problem.

In our experiments, for each problem, we put all of the corresponding first response time that are in the shortest 5% range for that problem into the first bin: the extremely short time bin; the student first response time within 5% to 50% range went into the second bin: the short time bin; the 50% to 95% range went into the third bin: the long time bin; and the top 5% went into the forth bin: the extremely long time bin. These four bins are denoted as bin1 to bin4 in our training dataset. This numbers 5%, 50% and 95% are selected based on experimenting with a few different sets of values. We did not try more sophisticated criteria, such as standard deviation, which might be able to further improve the result.

This method allows us to consider the main trend of the student response time per problem, without being affected by rare and extreme situations or data.

## 2.3 Predicting Student Performance

In this section, the purpose of our analysis is to find out if student first response time is valuable in modeling student knowledge and predicting student performance. We want to model only student first response time in this step, so that the result won't be affected by other additional features. Also, we want the model to be very simple so that it can be easily computed and leveraged into other existing student models that using other features for modeling student knowledge.

We choose to use a purely data driven tabling model that is similar to our previous work [4], which makes no assumptions about how the new information reflects student knowledge. To do so, we simply built a one by four parameter table, in which column index represents the category of student first response time in the previous question, and each cell contains the probability that the student will answer the current question correctly. For that value, we simply use the percentage of students who answered the current question correct when the previous question fell into the corresponding category.

Table 1 shows the parameter table we computed from the training data.

| Bin 1 | Bin 2 | Bin 3 | Bin 4 |
|---|---|---|---|
| 0.3829 | 0.7103 | 0.6428 | 0.5389 |

**Table 1. Parameter table computed from the training dataset.**

This model is very simple and easy to compute. But also, it is very limited. The only information it takes into account is the student first response time and the difficulty or the type of question. The information of the question is included in the model for when we discretized the first response time, we choose different bin cut points for different questions.

To evaluate how well this simple model fits the data compare to a baseline of always guessing the mean value of the data as a prediction. We used Root Mean Squared Error (RMSE) as a metric to examine the predictive performance on an unseen test set. The RMSE of the baseline prediction is 0.4589 and the RMSE of the student first response time model is 0.4552, which indicates this value is indeed contain some predictive power, although the benefit of this information is not obvious.

## 2.4 Leveraging First Response Time into KT

In this section, our goal is to find out whether or not leveraging the result of the simple model above into an existing student model which does not take into account student first response time information could help improve the existing student model, and thus result in better prediction accuracy. We choose the KT model in our experiments.

By combining the student first response time model with the KT model, we leverage new information into the KT model. To find out the result of this method, we used a linear regression model to combine the simple model we built with the traditional KT model by making the student performance as the dependent variable in the regression model, and the prediction results from the student first response model and the KT model as independent variables.

We again used the RMSE to examine the predictive performance of the KT model and the combination of these two models. The result is shown in Table 2. The FRT in Table 2 represents the first response time model, KT represents the Knowledge Tracing model, and the Comb represents the linear regression combination of these two models. This table also provides the comparison of the number of parameters of each model. Since the data set has 220 skills, KT generated in total 4*220 parameters.

| | FRT | KT | Comb |
|---|---|---|---|
| RMSE | 0.4552 | 0.4251 | 0.4213 |
| #of params | 4 | 880 | 886 |

**Table 2. Comparision of the RMSE result of different models.**

The linear regression formula for combining two models tells us the information about the weight of each model in regarding with their impact to the final model. The formula generated from our training process of the linear regression is:

-0.1227 + 0.1928 * FRT_prediction + 0.9821 * KT_prediction.

from which we can tell that the influence of the student first response time model to the final result is small. However, the RMSE shows an improvement from the KT model.

To find out if this improvement is statistically reliable, we did reliability analysis by computing the student level RMSE to account for the non-independence of each student and their actions and then compared the KT and the Comb model using a

two tailed paired t-test. The p value is 0.0389, which indicates that although the improvement is small, it is reliable.

## 3. CONTRIBUTIONS

This paper makes two main contributions. First, we analysed the predicting power of student first response time on student performance. In compare with other work on the student response time, which focus on explaining student in task or off task behavior, this work shows that student first response time contains certain information about student knowledge.

The second contribution this paper makes is to show that by leveraging the student first response time information, we can improve the prediction accuracy of the traditional KT model. In compare with other more complicated and time consuming methods, this model is very flexible and easy to apply to any existing student modeling techniques to incorporate into them the new information of student first response time.

## 4. FUTURE WORK AND CONCLUSIONS

The model we proposed for using student first response time to improve KT model is a simple and fast way of utilizing additional information. However, experiments show using student first response time alone did not provide a good performance prediction. There are several questions that we are interested in exploring.

One question is if the prediction accuracy of using student first response time can be improved by taking into account student and skill information. Currently we use only four parameters for all of the data. This can be easily extended to deal with individualization and separate skill by computing parameter tables for each skill or each student separately.

Another question we want to explore is a way to combine the response time and other information that gathered when a student answers a question, such as the number of hints and attempts a student need to answer the question. We are interested in combine these features because they seem to be highly related. We built a tabling model using the assistance student needs for answering a question in 2010[4], and searching for a method to merge these two models together is a reasonable next step.

In conclusion, in this paper, we use a method that is easy to compute and apply to leverage discretized student first response time information into the KT model to improve the prediction accuracy of the KT model. The result shows a clear value of student first response time in indicating student knowledge.

## ACKNOWLEDGMENTS

This research was supported by the National Science foundation via grant "Graduates in K-12 Education" (GK-12) Fellowship, award number DGE0742503 and Neil Hef-fernan's CAREER grant. We also acknowledge the many additional funders of AS-SISTments Platform found here: http://www.webcitation.org/5ym157Yfr

All of the opinions expressed in this paper are those solely of the authors and not those of our funding organizations.

## REFERENCES

[1] Corbett, A., Anderson, J. 1995. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. User Modeling and User-Adapted Interaction 4:253-278.

[2] Pardos, Z. A., Heffernan, N. T. (2011) KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model. In Proceedings of the 19th International Conference on User Modeling, Adaptation and Personalization. Girona, Spain.

[3] Qiu, Y., Qi, Y., Lv, H., Pardos, Z.A, Herffernan, N. T.: Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing, In Proceedings of the 4th International Conference on Educational Data Mining. 2011.

[4] Wang Y., Heffernan N. T., The "Assistance" Model: Leveraging How Many Hints and At-tempts a Student Needs, In Proceedings of the 24th International FLAIRS Conference. Palm Beach, Florida.

[5] Beck, J. E.: Using response times to model student disengagement, In Proceedings of AIED 2005.

[6] Shih, B., Koedinger K. R., Scheines, R.: A Response Time Model for Bottom-Out Hints as Worked Examples, In Proceedings of the 1st International Conference on Educational Data Mining, 2008, p. 117-126.

[7] Ivon Arroyo, Hasmik Meheranian, Beverly Park Woolf: Effort-based Tutoring: An Empirical Approach to Intelligent Tutoring. EDM 2010: 1-10

[8] Dovan Rai, Joseph Beck: Exploring User Data From a Game-like Math Tutor: A Case Study in Causal Modeling. EDM 2011: 307-313

# Meta-learning approach for automatic parameter tuning: A case study with educational datasets

M.M. Molina, J.M. Luna, C.Romero, S.Ventura
Department of Computer Science
University of Cordoba, Spain
i02mamom@uco.es, i32luarj@uco.es, cromero@uco.es, sventura@uco.es

## ABSTRACT

This paper proposes to the use of a meta-learning approach for automatic parameter tuning of a well-known decision tree algorithm by using past information about algorithm executions. Fourteen educational datasets were analysed using various combinations of parameter values to examine the effects of the parameter values on accuracy classification. Then, the new meta-dataset was used to predict the classification accuracy on the basis of the value parameters and some characteristics of the dataset. The obtained classification models can help us decide how the default parameters should be tuned in order to increase the accuracy of the classifier when using different types of educational datasets.

## Keywords

parameter tuning, classification, J48 algorithm

## 1. INTRODUCTION

One of the objectives of Educational Data Mining (EDM) [10] must be to design easy-to-use tools and algorithms for educators and non-expert users of data mining. Traditional data mining tools, such as Weka, Rapid-Miner, Clementine, DB-Miner, etc., are normally designed more for power and flexibility than for simplicity. Therefore, these tools can be complex, with features well beyond the scope of an educator's needs. Most current data mining algorithms used by these tools need to be configured before they are executed. In other words, users have to provide appropriate values for the parameters in advance in order to obtain good results or models; therefore, the user must possess a certain amount of expertise in order to find the right settings. To resolve this problem, data mining can be used to learn from past executions of the algorithms in order to improve the future selection of parameters according to the past behaviour of the algorithm.

In this paper, we propose a meta-learning approach for tuning parameters. Meta-learning is the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and the data mining process [1]. In our case study, we used a meta-learning approach to support the user in tuning the parameter values of a decision tree classification model when using different types of educational datasets. The decision tree model has some parameters that influence the amount of pruning. By trimming trees, the computational efficiency and classification accuracy of the model can be optimised. As a case study, we used a set of educational datasets and the J48 [9] (improved version of the C4.5 classification algorithm) to predict a discrete variable or class (accuracy variations) based on the values of the parameters and some features of the datasets. We executed some combinations of parameter values to examine their effects on a classification quality metric.

This paper is organised as follows: Section 2 provides background information from related works on applying data mining for parameter tuning; Section 3 describes the methodology used in this work; Section 4 includes the list of educational datasets used as a case study; Section 5 describes the experiments, results, and model obtained; and finally, conclusions and future works are outlined in Section 6.

## 2. BACKGROUND

In data mining, it is generally necessary to set the parameters used by the algorithm in order to achieve the best possible model and results [7]. Experiments show a substantial increase in accuracy when the right parameters are used. However, there is an associated problem in adjusting the parameters of most data mining algorithms. This task may involve a high computational cost for finding the optimal parameters or else risk relying on assumptions that may bias the results. Achieving optimal parameters automatically is not an easy task, therefore, and it often requires help from an expert. Some possible solutions include providing default values to the user (the most simple and common solution), reducing the number of parameters, tuning parameters automatically (the chosen option in this paper), and developing parameter-free data mining [6] algorithms (the ideal but most difficult solution).

The area of automatic parameter tuning research has gained much interest in recent years [13]. The definition of automatic parameter tuning used in this paper is to automatically find parameter settings that are better than the defaults. Different methods and techniques have been proposed for automatic parameter tuning [2], such as optimisation techniques (racing algorithms, local search, experimental design, etc.), machine learning and/or data mining. In fact, classifiers have been used to learn the values of parameters needed to set the configuration. Maimon, Rockach, and Edel [7] describe a classification model for meta-based parameter tuning. Srivastava and Mediratta [11] suggest the use of decision trees for automatic tuning of search algorithms. Pavon, Diaz, Laza, and Luzon [8], have automated the parameter tuning process through classification of previous runs of the algorithms. Dakovski and Shevked [3] consider an algorithm for learning from examples from the view point of improving classification accuracy by determining influencing parameters and optimal values.

This paper focuses on automatic parameter tuning by supporting the selection of the parameter values of a J48 classifier. The obtained model can help us make decisions about how we can tune the default parameters to increase the accuracy of the classification when using different types of educational datasets.

## 3. METHODOLOGY

We propose a methodology that uses a meta-learning approach to support the selection of parameter values for the algorithms (see Figure 1).

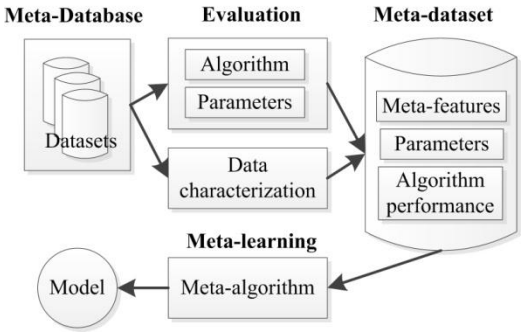



**Figure 1. Meta-learning approach**

In our meta-learning approach (see Figure 1), the meta-database consists of educational datasets. Then, we defined properties that are important for characterising datasets and developing meta-features (the number of instances, attributes, and classes). We selected a base algorithm, and parameters, to evaluate its performance. In this case, we selected the J48 algorithm and two parameters (*confidenceFactor* and *minNumObj*) to obtain the meta-dataset with meta-features, parameters, and performance (classification accuracy). Finally, meta-learning (a meta-algorithm) was applied to the previous meta-dataset in order to obtain a classification model for predicting whether an increase or decrease in estimated accuracy is to be expected for a given record. Each record of the meta-dataset represents a type of dataset and a certain parameter setting.

## 4. DATASETS

We used a set of 14 educational datasets based on the traditional classification problem for predicting students' final performance [10]. These datasets (see Table 1) contain as input attributes a variety of information about students and as classes (the output attribute to predict) the categorical final marks obtained by students in different types of courses:

- Moodle 1 to 7: Data about first, second, and third–year students for a degree in computer science at Cordoba University during the years 2007–2010, obtained from Moodle (accesses, assignments, and activities in questionnaires, forums, etc.)
- Higher 1 and 2: Data about first–year Cordoba students for a degree in computer science during 2010, obtained from several sources (admission and progress in subjects, Moodle, and a survey)
- Secondary 1 to 5: Data about students of secondary education in Zacatecas, Mexico, during 2010, obtained from several sources (admission information, scores in subjects, and a specific survey)

Table 1 shows the list of educational datasets and three features of these datasets: the number of attributes (Nattributes), the number of instances (Ninstances) and the number of classes (Nclasses). Clearly, there is a wide range of values in the features of each dataset. In fact, there are datasets with a low, medium, or high number of attributes, instances, or classes.

| Dataset | Nattributes | Ninstances | Nclasses |
|---|---|---|---|
| Moodle1 | 4 | 1000 | 5 |
| Moodle2 | 10 | 103 | 3 |
| Moodle3 | 41 | 103 | 3 |
| Moodle4 | 6 | 2708 | 3 |
| Moodle5 | 6 | 9554 | 3 |
| Moodle6 | 10 | 438 | 4 |
| Moodle7 | 10 | 438 | 2 |
| Higher1 | 24 | 88 | 6 |
| Higher2 | 24 | 88 | 2 |
| Secondary1 | 77 | 670 | 2 |
| Secondary2 | 14 | 670 | 2 |
| Secondary3 | 60 | 419 | 2 |
| Secondary4 | 17 | 386 | 2 |
| Secondary5 | 53 | 419 | 3 |

**Table 1. Features of the educational datasets**

## 5. EXPERIMENTS

Experiments were conducted to predict how to increase or decrease the accuracy of a well-known classification algorithm, depending on the parameters used and the features of the educational datasets used, using past information about algorithm executions. The decision tree learner selected was J48, which has several parameters but only two of which influence the amount of pruning [12]:

- ***confidenceFactor*** is the confidence factor for pruning, and it influences the size and predictability of the tree constructed. For each pruning operation, it defines the probability of error in the hypothesis that deterioration due to this operation is significant. The default value is 0.25. The lower this value, the more pruning operations allowed.
- ***minNumObj*** is the minimum number of instances per leaf. The default value is 2.

We executed the algorithms using different settings and stored the accuracy obtained in each execution as part of the meta-database. In fact, J48 was executed several times for each dataset by modifying these parameters into a range (in a similar way that an optimiser works). Each setting was evaluated using 10-fold cross-validation, and the accuracy (rate of correctly classified instances) obtained from test data was stored. The settings used were: *confidenceFactor* (0.1, 0.25, and 0.5) and *minNumObj* (1, 2, and 10), that is, a total of nine different combinations of parameters for each dataset. Next, in order to have a classification problem (that is, a class), we transformed the continuous value (float) of the obtained accuracy to a discrete or categorical value (label) in the following way:

- The accuracy value obtained when using the two default parameters together (0.25 and 2) was used as a control value; therefore, it was not discretised and was not used later for predicting (only the remaining eight executions).
- All the other accuracy values obtained were used as experimental values and transformed to the labels Equal, Increase, Decrease, Increase+, and Decrease− depending on

the variation of accuracy with respect to the control accuracy. In other words, each value was compared with the accuracy obtained using the default settings, and the label describes the difference: no difference (Equal), a higher or lower accuracy (Increase or Decrease, respectively), a much higher or lower accuracy (Increase+ or Decrease− respectively).

Finally, all the previous information was stored in a meta-dataset with 112 instances/examples and six attributes (five numerical attributes (three meta-features and two parameters) and one class (accuracy variation)). However, in order to create a different version of the same meta-dataset, we discretised all the numerical values. The labels used by *ConfidenceFactor* are LOWER to 0.1, DEFAULT to 0.25, and HIGHER to 0.5. The labels used by *MinNumObj* are LOWER to 1, DEFAULT to 2, and HIGHER to 10. The labels used by Nattributes, Ninstances, and Nclasses are shown in Table 2.

| | **LOW** | **MEDIUM** | **HIGH** |
|---|---|---|---|
| **Nattributes** | ≤10 | >10 AND ≤30 | ≥ 30 |
| **Ninstances** | ≤ 100 | >100 AND ≤1000 | > 1000 |
| **Nclasses** | = 2 | >2 AND ≤ 4 | > 4 |

**Table 2. Discretisation of the meta-features**

Based on the two previous meta-datasets, meta-learning (discrete and numerical classification) was used to predict the variation of the accuracy depending on the meta-features of the dataset and the values of the parameters. We used different types of classification algorithms provided by Weka [12]:

- Bayes-based algorithms: BayesNet, NaiveBayes
- Functions-based algorithms: Logistic, RBFNetwork, and MultilayerPerceptron
- Rules-based algorithms: JRip, NNge, PART, and Ridor
- Trees-based algorithms: LADTree, SimpleCART, REPTree,and J48

All these algorithms were executed using default parameters and 10-fold cross-validation, and their accuracy when using the original numerical attributes (A) was compared with their accuracy when using the categorical attributes (B) (see Table 5).

In general, none of the meta-learning classification algorithms obtained a very high accuracy, with values varying between 50% and 75% of correctly classified instances (see Table 3). From the results using original numerical attributes (column A) and those using categorical attributes (column B), it is apparent that all the algorithms obtained better results when using the original numerical attributes. Finally, the algorithm that obtained the highest accuracy in both cases (A and B) was the J48 classifier.

| **Algorithm** | **(A)** | **(B)** |
|---|---|---|
| BayesNet | 0.573 | 0.492 |
| NaiveBayes | 0.573 | 0.492 |
| Logistic | 0.617 | 0.573 |
| RBFNetwork | 0.617 | 0.537 |
| MultilayerPerceptron | 0.537 | 0.519 |
| JRIP | 0.573 | 0.528 |
| NNge | 0.671 | 0.492 |
| PART | 0.671 | 0.600 |
| RIDOR | 0.600 | 0.591 |
| LADTree | 0.671 | 0.582 |
| SimpleCart | 0.689 | 0.564 |
| REPTree | 0.635 | 0.573 |
| J48 | **0.751** | **0.698** |

**Table 3: Accuracy of classification algorithms**

Next, we describe the two classification models obtained by the J48 algorithm. These decision trees can easily be interpreted by a human and can help in making decisions about how to tune parameter values in order to increase the accuracy of the classification when using different types of datasets. Figure 2 shows part of the J48 pruned tree obtained when using the meta-datasets with numerical attributes.

```
Ninstances <= 103
|   Nclasses > 4
|   |   minNumObj <= 1: Decrease
|   |   minNumObj > 1: Increase+
|   Nclasses <= 4
|   |   Nattributes <= 17
|   |   |   minNumObj <= 2: Increase
|   |   |   minNumObj > 2: Decrease-
|   |   Nattributes > 17: Decrease-
Ninstances > 103
|   Ninstances <= 2708
|   |   minNumObj <= 2
|   |   |   Ninstances <= 386: Equal
|   |   |   Ninstances > 386
|   |   |   |   Nattributes <= 53
|   |   |   |   |   Nattributes <= 24
|   |   |   |   |   |   confidenceFactor <= 0.25
|   |   |   |   |   |   |   Ninstances <= 1000
|   |   |   |   |   |   |   |   Ninstances <= 438
|   |   |   |   |   |   |   |   |   minNumObj <= 1: Increase
|   |   |   |   |   |   |   |   |   minNumObj > 1: Equal
```

**Figure 2. Part of the decision tree using numerical attributes**

As we can see, all the input attributes (the three meta-features and the two parameters) appear in the decision tree; therefore, all show a relationship with the variations of accuracy. For example, the first two rules of the tree show that if the number of instances is less than 103 and the number of classes is greater than 4, then the value of the *minNumObj* parameter can decrease the accuracy a little (for a value less than or equal to 1) or can increases it quite a lot (for a value greater than 1).

Figure 3 shows part of the J48 pruned tree obtained using the meta-datasets with discrete attributes.

```
Ninstances = LOW
|   Nclasses = HIGH
|   |   minNumObj = LOWER: Decrease
|   |   minNumObj = DEFAULT: Increase+
|   |   minNumObj = HIGHER: Increase+
|   Nclasses = MEDIUM: Increase+
|   Nclasses = LOW: Decrease-
Ninstances = MEDIUM
|   Nclasses = HIGH
|   |   confidenceFactor = LOWER: Decrease
|   |   confidenceFactor = DEFAULT: Decrease
|   |   confidenceFactor = HIGHER: Increase
|   Nclasses = MEDIUM
|   |   Nattributes = MEDIUM: Increase
|   |   Nattributes = LOW: Increase
|   |   Nattributes = HIGH: Decrease-
```

**Figure 3. Part of the decision tree using categorical attributes**

As we can see in Figure 3, very similar rules are obtained and, again, all the input attributes appear in the decision tree. The three first rules of the tree show that if the number of instances is low and the number of classes is high, then the value of the *minNumObj* parameter can decrease the accuracy a little (for a value lower than the default value) or can increase it quite a lot (for a value equal to or higher than the default value). In our opinion, this second decision tree is a little more comprehensible to a human for two main reasons:

1. The tree is much smaller. The first decision tree (Figure 2) has 47 nodes and 24 leaves (rules), and the second decision tree (Figure 3) has 28 nodes and 19 leaves (rules). We maintain that a small decision tree with fewer and shorter rules is more comprehensible.

2. Although the accuracy of the classification is lower when discretising (see Table 3), the use of labels instead of numbers and operators (equal, greater than, less than, etc.) provides more simple rules. We maintain that a decision tree with labels or linguistic variables is more comprehensible.

# 6. CONCLUSIONS

In this paper, we have shown that a meta-learning approach can be used for parameter tuning of decision tree algorithms. We used 14 educational datasets because there are no more datasets on classification tasks in education available. Although there are some public and well-known data repositories, such as the UCI machine learning repository [4] and the PSLC DataShop [5], there are no educational datasets available in UCI and the PSLC datasets are oriented to predicting student step-level performances and not to the classification problem/task of predicting final marks. The ideal would be to use a great number of educational classification datasets from different types of education systems, such as primary, secondary, higher, special education, and so on, both in traditional face-to-face and in on-line education (learning management systems, adaptive educational hypermedia systems, intelligent tutoring systems, etc.). We selected the J48 algorithm and only two of its parameters, but in the future, other well-known algorithms and a great number of parameters may be used to broaden the research on the relationship between parameters and performance (accuracy). Finally, we used only three basic characteristics of the datasets (number of instances, number of attributes, and number of classes). However, future research may use other characteristics, such as level of missing data, level of imbalance in data, level of complexity, and so on.

# 7. ACKNOWLEDGMENTS

This work was supported by the Regional Government of Andalusia and the Spanish Ministry of Science and Technology projects, P08-TIC-3720 and TIN-2011-22408, respectively, FEDER funds, and the Spanish Ministry of Education under the FPU grant AP2010-0041.

# 8. REFERENCES

[1] Brazdil, P., Giraud-Carrier, C., Soares, C. and Vilalta, R. Metalearning: Applications to Data Mining. *Series: Cognitive Technologies*. Springer, 2009.

[2] Cayci, A., Eibe, S., Menasalvas, E. and Saygin, Y. Bayesian networks to predict data mining algorithm behavior in ubiquitous computing environments. *International Workshops MSM*, 2010.

[3] Dakovski, L. and Shevked, Z. Tuning classification for prime implicants based learner. *International Conference on Computer Systems and Technologies*, 2006.

[4] Frank, A. and Asuncion, A. UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science, 2010.

[5] Koedinger, K., Cunningham, K., Skogsholm, A. and Leber, B. An open repository and analysis tools for fine-grained, longitudinal learner data. *1st International Conference on Educational Data Mining*, 157–166, 2008.

[6] Keogh, E., Lonardi, S. and Ratanamahatana, C.A. Towards parameter-free data mining. *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 206–215, 2004.

[7] Maimon, O., Rokach, L. and Edel, I. Parameter tuning for classification algorithms in data mining using meta learning. *13th Israeli Conference of Industrial Engineering and Management*, 2004.

[8] Pavon, R., Diaz, F., Laza, R. and Luzon, V. Automatic parameter tuning with a Bayesian case-based reasoning system. A case of study. *Expert Systems with Applications*, 36(2), 3407–3420, 2009.

[9] Quinlan, J. R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[10] Romero, C. and Ventura, S. Educational Data Mining: A Review of the State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(6), 601–618, 2010.

[11] Srivastava, B. and Mediratta, A. Domain-dependent parameter selection of search-based algorithms compatible with user performance criteria. *Proceedings of AAAI*, 3, 1386–1391, 2005.

[12] Witten, I. H., Eibe, F. and Hall, M.A. *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*. Morgan Kaufman Publishers, 2011.

[13] Konen, W., Koch, P., Flasch, O., Bartz-Beielstein, T., Friese, M. and Naujoks, B. Tuned data mining: a benchmark study on different tuners. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 1995–2002, 2011.

# Mining Concept Maps to Understand University Students' Learning

Jin Soung Yoo
Indiana University-Purdue University, Fort wayne
2101 Coliseum Blvd East,
Fort Wayne, Indiana 46805
yooj@ipfw.edu

Moon-Heum Cho
Kent State University, Stark
6000 Frank Avenue, N. W.
North Canton, Ohio 44720
mhcho@kent.edu

## ABSTRACT

Concept maps, visual representations of knowledge, are used in an educational context as a way to represent students' knowledge, and identify mental models of students; however there is a limitation of using concept mapping due to its difficulty to evaluate the concept maps. A concept map has a complex structure which is composed of concepts and their relationships that often have a weighted direction. This work explores the feasibility of the analysis of concept maps using data mining methods, and investigate the possibility of using concept maps as a research tool to understand college student's learning. A total of 111 college students participated in this study. The findings from frequent concept mining and sub-concept map mining suggest that students expect a traditional way of learning. The study also shows a promising area of further study in the area of data mining in education.

## 1. INTRODUCTION

Understanding how students learn is an important issue in college education [14; 15]. Perhaps, the most common methods to be used to understand learning of college students were both interviews [14] and self-reporting measurements such as Learning and Study Strategies Inventory (LASSI) [20] or Motivated Strategies for Learning Questionnaire (MSLQ) [15]. Recently, however, researchers agreed upon the importance of alternative methods to understand students' learning [21; 24].

Concept maps which are visual representations of knowledge are used in educational contexts, as a way to represent students' knowledge, and identify mental models of students [13]. It includes concepts, usually enclosed in circles or boxes of some type, and relationships between concepts indicated by a connecting line that links two concepts [12]. Figure 1 shows an example of concept map. The map depicts the way a student studies or prepares for an exam. Using nodes and links, students can visualize their ideas and understanding of the content. Because the concept maps represent students' mental models of content and their ways of problem solving, concept mapping has been widely used in teaching and learning process in academia ranging from K-12 to higher education [7; 11].

Although concept mapping is a powerful tool, there is a limitation of using concept mapping due to its difficulty to evaluate the concept maps.

The most common method to evaluate students' concept maps in educational setting might be using rubrics; however, this also has limitations. First, although educators may have a general understanding of their students' mental models from using rubrics, educators are not provided specific and detailed information about the relationships among the concepts. Second, although rubrics reduce the amount of time educators spend evaluating concept maps, evaluation is still very time-consuming using the concept maps, particularly, in a large classroom where students number over 50. Third, when using rubrics, evaluation may not be easy for the educators to determine certain patterns of the students' mental models. Each person's concept map will look differently, although they may have similar patterns. Consequently, there is a strong need of computerized methods to evaluate students' concept maps.

In educational science, Yin et al. [23] examined the equivalence of two construct-a-concept-map techniques: construct-a-map with created linking phrases and construct-a-map with selected linking phrases. Some works [10; 18] explored different scoring systems for concept maps. In educational engineering, several studies [5; 19; 4] describe the construction of concept maps. Alves et al. [2] presents a system called TextStorm, which extracts raw concept maps from text. Shen et al. [16] works on the automatic generation of concept maps through text mining techniques. These studies concern the generation of concept maps rather than the analysis of concept maps.

Data mining is the process of automatically extracting new and useful knowledge hidden in large datasets. Performing data mining on concept maps can provide useful information in understanding the thought processes which generate

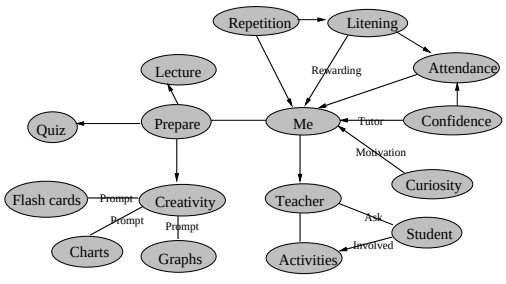



Figure 1: An example of concept map

them. In this study, we explore the feasibility of the analysis of concept maps using data mining methods, and investigate the possibility of using concept maps as a research tool to understand college student's learning.

## 2. DATA COLLECTION

We first describe participants in this study, concepts provided to them, and data collection procedure.

### 2.1 Participants

Data were collected from 10 college Critical Inquiry (CI) courses designed to help underprepared students to either acquire study skills or efficacy/motivation. The CI courses was chosen due to the large enrollment members, of approximately 300 students per semester, and they provide specific learning contexts through pairing the CI courses with other subject courses such as psychology, chemistry, biology, education, and archeology. The primary purpose of pairing the CI courses with other subjects courses is to facilitate student to learn better in a specific subject area. A total of 111 students participated in the study by drawing their concept maps. Based on the information students provided, 58 students were females, most students were Caucasian (n=83), and most students were freshmen (n=94).

### 2.2 Concepts

A total of 112 concepts were designed for this study. We reviewed the concepts with the two instructors who taught the study skill course. The 112 concepts consist of seven categories including classroom learning (e.g., listening, reading, or discussion, total 41 concepts), action for study (e.g., self-explanation, annotations, or memorization, total 18 concepts), learning tools (e.g., notes, charts, or textbooks, total 22 concepts), internal trigger (e.g., inquiry, curiosity, or creativity, total 4 concepts), motivation (e.g., will, confidence, or inspire, total 12 concepts), school facilities (e.g., library, web, or writing center, total 7 concepts), or people (e.g., teacher, classmate, or parent, total 8 concepts).

### 2.3 Procedures

Two weeks after mid-term exam week, we visited each class and administered the concept maps to students. In each class, we provided a 10-minute orientation about concept maps and drew one map for students on a whiteboard as a demonstration. Students were asked to respond to the question "*how do you learn in a college class?*". They drew their concept maps with pen and pencil since the class room was a regular class room that didn't have computers for individual students.

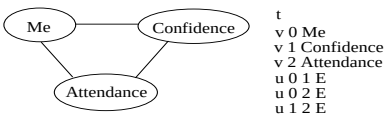
## 3. CONCEPT MAP MINING

Through discussions with education researchers, we listed their interesting queries to the students' concept maps. In this work, we focused on two problems: "Which concepts are frequently used in students' concept maps?" and "Which sub concept structures are commonly observed in students' concept maps?". For answering the first question, we conducted frequent item set mining task [1] to the concept map data, and for the second question, we used sub-graph mining task [9].

### 3.1 Data preprocess

| TID | Items |
|---|---|
| 1 | {Repetition, Listening, Attendance, Me, Confidence, Teacher} |
| 2 | {Prepare, Lecture, Me, Quiz, Teacher, Flash cards, Charts} |
| 3 | {Assignments, Study, Friend, Teacher, Notes, Activities} |
| 4 | {Quiz, Prepare, Lecture, Creativity, Me, Teacher, Time, Note} |
| ... | ... |

(a) Transaction data




(b) Graph data

Figure 2: Input data formats for data mining

After collecting students' hand-drawn concept maps, we digitalized them. Concept maps can be represented by graphs, consisting of nodes( or vertices), which represent concepts, and arcs( or edges), which represent relationships between the concepts. In the graphs of concept maps, vertices should have labels associated with them. Edges may have associated labels and directions represented by arrows. Vertices and edges may have their own weighted value to designate their significance.

Although a list of predefined concept names and relationship names were provided, students misspelled some names or gave wrong concept and relationship names different from the predefined names. We also noticed, that in several instances synonyms or plurals of the same word were used as labels, and that some students did not follow a standard in the labeling of nodes and edges or in the use of arrows to denote direction on the edges. Our digitalization process checked all inconsistent concept names and relationship names in the concept maps.

### 3.2 Frequent association concept mining

In order to find common concepts students have used for their concept maps, we applied a methodology known as association analysis to the concept data. Association analysis task in data mining is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships are represented in the form of association rules or a sets of frequent items [17; 1]. For example, a frequent item set, {Lectures, Notes} suggests that a strong relationship exists between lecture and note in learning strategy. The frequency of the associated items is often measured with support. The *support* of an item-set $x$ is defined as the fraction of all transactions that contain $x$, i.e., $s(x)=\frac{sc(x)}{N}$, where $N$ is the total number of transactions, and $sc(x)$ is the *support count* of $x$, $sc(x) = |\{t_i | x \subseteq t_i, t_i \in T\}|$. If the support of item-set $x$ is greater than a given support threshold, $x$ is called a frequent item-set.

For the association analysis, data should be prepared with transaction data format. We transformed our digitalized concept map data to concept transaction data as shown in Figure 2 (a). Each row in this table corresponds to a transaction that contains a unique identifier labeled TID, and a set of concepts used by a student in drawing his/her concept map. Here, a relationship between two concepts is not included in the transaction. There are many algorithms for association analysis [1; 3; 17]. We used Apriori algorithm [1]

for our analysis. Four different minimum frequency thresholds 30%, 40%, 50% and 60% were used.

### 3.3 Frequent sub-concept map mining

The second analysis is performed in order to derive a set of common sub structures among the collection of concept graph data. We applied frequent sub-graph mining task [17; 6] to our concept map data. Each concept map can be represented as a graph data as shown in Figure 2 (a). A concept graph $G = (V, E)$ is composed of a concept vertex set V and a set of edges E connecting between pairs of vertices. A graph $G' = (V', E')$ is a sub-graph of another graph $G = (V, E)$ if its concept vertex set $V'$ is a subset of $V$ and its edge set $E'$ is a subset of $E$. The frequency of a sub-graph is also measured by support. The support for a sub-graph $g$ is defined as the fraction of all graphs that contain $g$ as its sub-graph, i.e., $s(g)=\frac{sc(g)}{|GD|}$, where $GD$ is a collection of graphs, and $sc(g)$ is the support count of $g$, i.e, $sc(g)=|\{G_i|g \subseteq G_i, G_i \in GD\}|$. Sub-graphs (sub-concept map structures) $g$ such that $s(g) \geq minsup$ are frequent sub-graphs. There are several algorithms for frequent sub-graph mining such as FSG [9], gSpan [22] and SPIN [8]. We fed the concept graph data to the FSG algorithm [9]. Figure 2 (b) shows an example of input data format for the FSG algorithm. The required order for a valid graph data begins with '$t$' followed by all vertexes in the graph and finally by all the edges. Three different frequency thresholds, 10%, 20% and 30% were used for our analysis.

## 4. RESULTS

107 concept maps among 111 maps were analyzed. Four concepts maps were deleted because the concepts were not recognizable or students did not follow the instructions. Given the 112 concepts, a total number of 110 concepts were used. The maximum number of concepts and lines a student used were 39 and 31 respectively in each concept map, and the average concepts and lines students used were 13 and 14 respectively.

Given 112 concepts, only 15 concepts: 'teachers', 'me', 'notes', 'lectures', 'time', 'repetition', 'textbooks', 'listening', 'flashcard', 'parents', 'reading', 'annotation', 'study', 'tutor' and 'activity' concepts are used by students with 30% frequency. Among association patterns having two concepts, {teachers, me}, {teachers, notes} and {me, notes} showed very high frequency of over 60%. Among patterns having three concepts, {teachers, me, time}, {teachers, lectures, repetition} and {teachers, textbooks, repetition} showed over 50% frequency. Two concept sets having four items, {teachers, lectures, repetition, time} and {teachers, textbooks, repetition, time} showed around 30% frequency.

In the sub-graph mining, the size of the patterns is measured by the number of edges. Table 1 shows sub-concept map structures with at least 10% frequency. When the threshold was set to 30%, three types of frequent sub-concept map patterns were found such as "me-listening"(sc = 34 out of 107), "teacher-lectures"(sc = 44), and "me-notes"(sc = 49). When the threshold comes to 20%, a total of 15 frequent sub-concept map structures were observed. Among them, 12 sub-graph structures were about 'me' strategies or actions, 2 sub-graphs were about 'teacher' related patterns, and 1 sub-graph was between me and teacher. Last, when the threshold was 10%, a total of 47 simple sub-graphs were discovered where 36 sub-graphs were 'me' initiated concepts, 6 sub-graphs were 'teacher' related concepts, 5 sub-graphs explain between teacher and me, with one or more of the other elements.

## 5. DISCUSSION

In this study, it was found that most students learn through 'me' initiated behaviors (e.g., "me-listening" or "me-ask") or using learning tools (e.g., "me-notes" or "me-textbooks"). Learning through active interaction with teachers or peers was not observed. This indicates that early level college students focus on individual learning rather than learning through interaction with others. It may also reflect lecture oriented college courses. However, readers should consider that too many concepts given to students may have resulted in difficulty identifying patterns between students and others.

From this initial research, we found both possibilities and challenges when using concept maps in this capacity in an educational context. First, when used with data mining techniques, concept maps can be useful to interpret large sets of concept maps. As can be seen in our data analysis,

| Relevant concept | Frequency (Support count) | Frequent sub-concept map patterns |
|---|---|---|
| Me(Students) | 49 | Notes - Me |
| | 34 | Listening - Me |
| Teachers | 44 | Lectures - Teachers |
| Me(Students) | 31 | Teachers - Me |
| | 30 | Textbooks - Me |
| | 29 | Flashcards - Me |
| | 29 | Lectures - Me |
| | 29 | Reading - Me |
| | 28 | Study - Me |
| | 27 | Attendance - Me |
| | 24 | Annotation - Me |
| | 21 | Pay attention - Me |
| | 22 | Flashcard - Me -Notes |
| Teachers | 22 | Notes - Teachers |
| Etc. | 21 | Teacher - Lecture - Me |
| Me(Students) | 16 | Questions - Me |
| | 16 | Review - Me |
| | 16 | Discussions - Me |
| | 15 | Assignments - Me |
| | 14 | Study - Me |
| | 14 | Memorization - Me |
| | 13 | Classmates - Me |
| | 12 | Asking - Me |
| | 12 | Class - Me |
| | 18 | Lecture - Me - Notes |
| | 16 | Listening - Me - Reading |
| | 15 | Reading - Me - Notes |
| | 15 | Attendance - Me - Notes |
| | 14 | Notes - Me - Teacher |
| | 14 | Annotation - Me - Notes |
| | 13 | Study - Me - Notes |
| | 12 | Listening - Me - Notes |
| | 11 | Textbooks - Me - Flashcard |
| | 11 | Annotation - Me - Textbook |
| | 11 | Reading - Me - Teacher |
| | 11 | Lecture - Me - Textbooks |
| | 11 | Attendance - Me - Textbooks |
| | 11 | Listening - Me - Pay attention |
| Teachers | 15 | Discussion - Teacher |
| | 14 | Questions - Teacher |
| | 13 | Teach - Teachers |
| | 12 | Listening - Teacher |
| Etc. | 18 | Teacher - Notes - Me |
| | 13 | Lectures - Teacher - Me |
| | 11 | Teacher - Questions - Me |
| | 12 | Teacher - Lecture - Me - Note |

Table 1: Frequent sub concept map patterns

when using different thresholds, we were able to find certain patterns of students' learning in a college classroom. Although most of the patterns were simple such as "me-listening" or "me-annotations", this does not mean the concept mapping was an ineffective method to capture sophisticated pictures of students' learning. The simple patterns can be attributable to the large list of 112 predesigned concepts. Because students had too many choices, students' concept maps became very diverse. However, through this initial phase of study, we were able to find the most frequently referenced concepts for future study. With limiting the number of concepts, e.g., 30 concepts, we may be able to find more detailed pictures of college students' learning. In our future study, we will compare and contrast patterns of concept maps to students' self-report motivation or metacognition. For example, we can divide students into several groups based on their self-report questionnaires, and then compare how the patterns of concept maps are similar or different between groups. We are also planning to use students' final grades as a way to validate the patterns of learning. The study will be appealing to those who are interested in concept maps as an alternative tool for research as well as data mining in education.

## 6. REFERENCES


[1] R. Agarwal and R. Srikant. Fast algorithms for Mining association rules. In *Proc. of International Conference on Very Large Databases(VLDB)*, 1994.

[2] A. O. Alves, F. C. Pereira, and A. Cardoso. Automatic reading and learning from text. In *Proc. of International Symposium on Artificial Intelligence*, 2002.

[3] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proc. of ACM SIGMOD International Conference Management of Data*, 1997.

[4] N.-S. Chen, P. Kinshuk, C.-W. Wei, and H.-J. Chen. Mining e-learning domain concept map from academic articles. *Computers & Education*, 50:694-698, 2008.

[5] R. B. Clariana and R. Koul. A computer-based approach for translating text into concept map-like representations. In *Proc. of International Conference on Concep Mapping*, 2004.

[6] D. Cook and L. Holder. *Mining graph data*. Wiley Publication, 2007.

[7] K. en Chang, Y.-T. Sung, and I.-D. Chen. The effect of concept mapping to enhance text comprehension and summarization. *The Journal of Experimental Education*, 71(1), 2005.

[8] J. Huan, J. Prins, W. Wang, and J. Yang. SPIN: Mining maximal frequent subgraphs from graph databases. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[9] G. Karypis and M. Kuramochi. Frequent subgraph discovery. In *Proc. of International Conference on Data Mining*, 2001.

[10] I. M. Kinchin. Using concept maps to reveal understanding: A Two-tier analysis. *School Science Review*, 81:41-46, 2000.

[11] J. R. McClure, B. Sonak, and H. K. Suen. Concept map assessment of classroom learning: Reliability, validity, and logistical practicality. *The Journal of Research in Science Teaching*, 36:475-492, 1999.

[12] J. D. Novak and A. J. Canas. The theory underlying concept maps and how to construct and use them. Technical report IHMC Cmap Tools 2006-01 Rev 01-2008.

[13] J. D. Novak and D. B. Gowin. *Learning how to learn*. Cambridge. UK: Cambridge University Press, 1984.

[14] W. G. Perry. *Forms of intellectual and ethical development in the college years: A scheme*. New York:Holt, Rinehart & Winston, 1970.

[15] P. R. Pintrich, D. A. Smith, T. Garcia, and W. J. McKeachie. Reliability and predictive validity of the Motivated Strategies for Learning Questionnaire (MSLQ). *Educational and Psychological Measurement*, 53:801-813, 1993.

[16] R. Shen, R. Richardson, and E. Fox. Concept maps as visual interfaces to digial libraries: summarization, collaboration, and automatic generation. In *http://vw.indiana.edu/ivira03/shen-et-al.pdf*, 2004.

[17] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addision Wesley, 2005.

[18] E. M. Taricani and R. B. Clariana. A technique for automatically scoring open-ended concept maps . *Educational Technology Research & Development*, 54:65-82, 2006.

[19] A. Valerio and D. Leake. Jump-starting concept map construction with knowledge extracted from documents. In *Proc. of International Conference on Concept Mapping*, 2006.

[20] C. E. Weinstein, A. C. Schulte, and D. R. Palmer. *LASSI: Learning and study strategies inventory*. H & H Publishing Co., 1987.

[21] P. Winne and N. Perry. *Measuring self-regulated learning. In M. Boekaerts, P. Pintrich, & M. Zeidner (Eds.) Handbook of self-regulation (532-566)* . CA: Academic Press, 2000.

[22] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *IEEE International Conference on Data Mining*, 2001.

[23] Y. Yin, J. Vanides, M. A. Ruiz-Primo, C. C. Ayala, and R. J. Shavelson. Comparision of two concept-mapping techniques: implications for scoring, interpretation, and use. *Journal of Research in Science Teaching*, 2:166-184, 2005.

[24] B. Zimmerman. Investigating self-regulation and motivation: Historical background, methodological developments, and future prospects. *American Educational Research Journal*, 45:166-183, 2008.

# Policy Building – An Extension To User Modeling

Michael V. Yudelson
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213, USA
yudelson@cmu.edu

Emma Brunskill
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213, USA
ebrun@cs.cmu.edu

## ABSTRACT

In this paper we combine a logistic regression student model with an exercise selection procedure. As opposed to the body of prior work on strategies for selecting practice opportunities, we are working on an assumption of a finite amount of opportunities to teach the student. Our goal is to prescribe activities that would maximize the amount learned as evaluated by expected post-test success. We evaluate the proposed approach using an existing dataset where data was collected performing random skill selection. Our results cautiously support the hypothesis that using policies designed to optimize the post-test score associated with higher learning outcomes, but more work is needed.

## 1. INTRODUCTION

Recently there has been significant interest in logistic-regression based student modeling methods, including Performance Factors Analysis [3], Instructional Factors Model [2], and Contextual Factors Analysis [4]. Such models can flexibly incorporate skill difficulties and individualized student parameters. There is evidence that such models outperform Knowledge Tracing in terms of predicting student performance [2]. However, to our knowledge there has been no work that uses such student models with instructional decision making about what skills students should practice or what activity to perform next to maximize learning.

For example, consider selecting between the following problems when teaching a student least common multiples:

*1 (Product). Sally visits her grandfather every 2 days and Molly visits him every 7 days. If they are visiting him together today, in how many days will they visit together again?*

*2 (LCM). Sally visits her grandfather every 4 days and Molly visits him every 6 days. If they are visiting him together today, in how many days will they visit together again?*

Problem 1 can be solved by simply multiplying the given numbers (hence the tag *Product*). Problem 2 is an *LCM* and multiplication will not work. An open question is which problem type should be selected, and at what point in the student's learning progress. The seemingly obvious approach of presenting the easier *Product* problem earlier, and the harder *LCM* later on may not be best as emphasis on the use of a partial strategy of solving problems on least common multiples could lead to learning misconceptions. However, starting with harder *LCM* problems too early could be too challenging and might delay learning. In addition, it is likely that which activity to choose should depend on the student's current understanding and student ability.

In this paper we consider automatically selecting among such problems based on an online estimate of the student's probability of getting these problems correct. Our work differs from work on strategies for selecting practice opportunities (or more generally, pedagogical activities) to help the student reach mastery. Instead in our work we assume that the objective is to select a fixed number of activities to give to the student in order to maximize the amount learned, as evaluated by expected post-test success. This may be a useful objective in some classroom settings where a fixed amount of time is available.

One important challenge when considering new methods for problem selection is how to evaluate these methods. Typically student tutoring data is collected using a fixed policy for selecting problems, and if the proposed new policy differs from the prior policy, it can be hard to evaluate it using the prior dataset. In this work we leverage an existing dataset where part of the data was collected by performing random skill selection. This allows us to evaluate the policies we compute by finding existing examples in the dataset that happen to match the proposed policy. We can then compare the empirical performance of the matching examples to the performance of the students' whose policy did not match the proposed policy. In this way we can use existing randomized data to perform a *post-hoc* analysis of alternate policy strategies that can be used.

Though the size of our data prevents any strong conclusions, our preliminary results are promising. They suggest that selecting policies designed to optimize the post-test score are associated with higher post-test scores than other policies. Further work is required to examine this in more detail.

## 2. APPROACH

We now describe how we model student learning, and then describe how we use these models to create adaptive policies for what activity to select.

### 2.1 Student Modeling

We use the Contextual Factors Analysis (CFA) [4] framework to model student learning. CFA is an educational data mining model. It was developed as an elaboration on a series of other cognitive models, namely Performance Factors

**Algorithm 1** BestNextSkill

**Input:** for student $i$, no. successes on skill 1 and 2, $s_{i1}, s_{i2}$, and no. failures $f_{i1}, f_{i2}$, all other parameters $\psi_i$, no. problems given $d$, no. problems left to give $D$
**Output:** expected post-test score *score* for best skill to practice, *bskill*
**if** $d = D$ **then**
  $score$=0;
  **for** $j = 1 : 2$ **do**
    {predict post-test score}
    $score = score + p(c|skill_j)$
    $skill = NULL$ {No more time to practice}
  **end for**
**else**
  **for** $j = 1 : 2$ **do**
    $f'_{ij} = f_{ij} + 1$ {practiced skill $j$, failed}
    $fscore_j^{fail}$=BestNextSkill(for $k \neq j : \langle s_{ik}, f_{ik} \rangle$, $s_{ij}$,$f'_{ij}$,$\psi_i$,d+1,D)
    $s'_{ij} = s_{ij} + 1$ {practiced skill $j$, success}
    $fscore_j^{suc}$=BestNextSkill(for $k \neq j : \langle s_{ik}, f_{ik} \rangle$, $s'_{ij}$,$f_{ij}$,$\psi_i$,d+1,D)
    $score(j) = p(f|s_{i1}, s_{i2}, f_{i1}, f_{i2}, \psi_i) * fscore_j^{fail} + p(s|s_{i1}, s_{i2}, f_{i1}, f_{i2}, \psi_i) * fscore_j^{suc}$
  **end for**
  $score = \max_j score(j)$
  $bskill = \arg\max_j score(j)$
**end if**

Analysis model [3], Additive Factors Model (AFM) [1], and Rasch 1PL IRT model [6]. In addition to account for the number of correct and incorrect attempts to apply a skill separately (as PFA does in contrast to AFM), it captures transfer effects of prior attempts with one skill on the other. A logistic regression form of CFA is given in Equation (1):

$$logit(p_{ij}) = \theta_i + \sum_{a \in Q_j} (\beta_a + \gamma_a s_{ia} + \rho_a f_{ia}) + \sum_{b \notin Q_j} (\gamma_b s_{ib} + \rho_b f_{ib}) \quad (1)$$

Here, $p_{ij}$ is probability that student $i$ solves problem $j$ correctly, $\theta_i$ is student's ability parameter, and $Q$ is a so-called Q-matrix [5] that encodes what skills are associated with $j^{th}$ problem (or a problem step). $\beta_a$, $\gamma_a$, and $\rho_a$ are complexity, success learning rate, and failure learning rate respectively; they pertain to the skill(s) that is (are) addressed in $j^{th}$ problem (or problem step). $\gamma_b$, and $\rho_b$ are success and failure transfer rates respectively; they capture transfer from skill $b$ to skill $a$. $s_{ix}$ and $f_{ix}$ are the number of prior success and failures with $x^{th}$ skill. In our prior work with CFA (rf. [4]) we found it to be superior to PFA, whether or not the transfer parameters ($\gamma_b$ and $\rho_b$ ) were significant. It is due to these reasons that we used CFA.

## 2.2 Adaptive Instructional Policies

We now consider how to use our student model to automatically create adaptive instructional policies. Consider the scenario where we have 2 different skills we would like the student to learn, and we have a fixed number of opportunities $D$ when we can give the student practice on either skill. We assume as input we are provided the CFA student learning parameters. The objective is to compute an adaptive policy for $D$ skill opportunities should be provided to the student in order to maximize his expected post-test performance on 1 question per skill. The policy computed is an adaptive, conditional policy, because it depends on the responses made by the student: as the student responds to each practice opportunity, we update the number of success and failures of the student over each skill. This in turn will change what is the next best skill practice opportunity to give to the student. The way we compute the policy can be thought of as constructing a forward search tree, where we alternatively consider all possible skill practice opportunities to provide next, and then the possible responses (success or failure) of the student. We repeat this expansion for the desired number of $D$ practice opportunities. At the end of this, at a tree leaf, we compute the expected post-test performance, given the successes and failures of the tree path to this leaf. This simply involves predicting the probability that the student will get a question about skill 1 correct plus the probability they will get a question about skill 2 correct. Both these quantities can be computed using the student model. We repeatedly take expectations and maximizations to use these leaf scores to decide what skill should be practiced at the current student state: see Algorithm 1 for details.[1] Two-steps of a sample adaptive policy are shown in Figure 2.1.

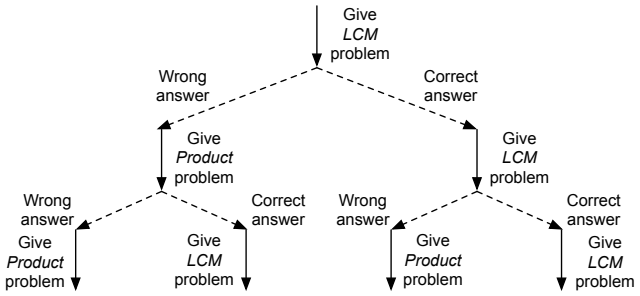



Figure 1: Example adaptive instructional policy.

Note that the computed "optimal" policy that is expected to maximize the student's post test performance is a direct function of the input student parameters. Therefore, the optimal policy can be different for different students.

# 3. DATA

The data comes from an experimental study conducted at Pinecrest Academy Charter Middle School. Students from $6^{th}$ and $7^{th}$ grades were exposed to a modified Carnegie Learning Bridge to Algebra (BTA) tutor. The part of the experiment we analyzed consisted of 10 sessions. In each of the sessions students were given 16 problems randomly drawn from a pool of 24 without replacement. One of the experimental conditions only included 8 problems to be delivered and it was removed for the sake of uniformity. Each session addressed a separate topic. Within a topic there were two or four skills, and the problems covered one or two of them. For example, one session was on least common multiples, and the skills were divided by: 1) whether the problem was formulated as a story or not ("story" or "word" problems), and 2) whether a solution can be obtained by mere multiplication or not ("product" and "true least common

[1]An alternative, but equivalent, method is to have Algorithm 1 return a complete conditional policy tree showing what skill to give after each possible student response.

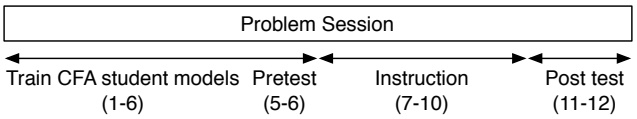



Figure 2: A topic session of 12 problems was divided into sections that we used to fit student models and consider pre and post performance after a period of 4 problems.

multiple" problems). In our analysis we group problems so that we considered only 2 alternate skills at a time.

# 4. EXPERIMENT

To evaluate our approach, we segmented each student's session data as follows (cf. Figure 3). Problems 1-6 were used to train the CFA models. These models were used to compute the instructional policy for a student to maximize expected post test score after doing 4 problems. The student's performance on problems 5-6 were used as a pretest score, then problems 7-10 were considered the tutoring/instructional phase, and the student's performance on problems 11-12 were considered a post test. Recall that the problems were selected randomly in the dataset that we used. We only used the first 12 problems (with a 4 problem "instructional" period) so that we could increase the likelihood of finding some overlaps in the data with the computed optimal 4-problem adaptive policies. Therefore we selected the subset of students who happened to get 1 problem for each of the 2 skills we considered in both the pretest and the post-test.

For comparison we also considered two alternate policies. One policy is to always give the student a problem for the skill that the student is more likely to solve correctly. We will call this policy an "easier problem" policy or just an "easy" policy. Our second comparison policy is to always provide the student with a problem that is for the skill that the student is less likely to solve correctly. We will call this policy a "harder problem" instructional policy or a "hard" policy. This harder policy is very similar to a common instructional approach used in Knowledge Tracing mastery learning in which a student is given an exercise for a skill that the student is least likely to have mastered.

We will compare the learning gains of students whose provided problems happened to match the 3 policies of interest (optimal, easy, or hard).

# 5. RESULTS

**Data restriction.** We focused our attention on the subset sessions where students improved between the pre- and post-test trials. The summary of learning effects between pre- and post-test trials is given in Table 1. Some sessions are listed twice (sessions 1, 3, and 6) because they contained multiple skills that will be divided into groups (e.g. Story-Word vs. Product-LCM in session 1). Sessions 5, 8, 9, and 10 were not considered because they contained errors in the data. We excluded sessions 2, 3 (both 3.1 and 3.2 versions), and 6.1 because students did not make measurable learning gains.

**Policy Performance.** A summary of the results of computing optimal policies for the students is given in Table 2. Recall that we compute an optimal policy for each student based on their student parameters. We then find instances in the data where the provided problems happened to match

Table 1: Learning between pre-test (trials 5 and 6) and post-test (trials 11 and 12)

| Session | No. stud. | Mean pre-test score | Mean post-test score | Learn. effect size | Learning t-test p-val |
|---|---|---|---|---|---|
| 1.1 | 48 | 1.06 | 1.52 | 0.73 | 0.000*** |
| 1.2 | 61 | 1.10 | 1.43 | 0.44 | 0.004** |
| 2 | 51 | 1.76 | 1.80 | 0.09 | 0.299 |
| 3.1 | 60 | 0.93 | 0.93 | 0.00 | 0.500 |
| 3.2 | 47 | 1.00 | 1.09 | 0.11 | 0.280 |
| 4 | 44 | 1.23 | 1.55 | 0.45 | 0.009** |
| 6.1 | 53 | 0.98 | 1.21 | 0.29 | 0.038* |
| 6.2 | 57 | 0.86 | 1.07 | 0.29 | 0.035* |
| 7 | 44 | 1.41 | 1.77 | 0.59 | 0.002** |

the optimal policy we computed. We repeat this process with the easy policy and the hard policy. Note that it is quite unlikely that the randomly selected problems will happen to match any of the 3 policies. Therefore it is not surprising that the number of matches we find in the data for each of the 3 policies is quite low, ranging from 1 to 14 for optimal policies and from 0 to 7 for comparison policies. Table 2 also lists number of students that follow overlaps of optimal and ad hoc policies.

The last 5 columns of Table 2 show the comparison between students that received a particular policy versus all other students. Though we caution against making sweeping claims because the number of students that followed any of the policies is very low, there remain some encouraging results. First, for session 1.1 and 1.2, students that received the optimal policy did better than than students that did not. The results were not significant, but trending that way (paired t-test p-value=0.090). In the other 3 sessions it is extremely difficult to assess any trends, as there were very few students that followed any policy at all.

It is not yet clear if optimal policies are significantly better than the comparison policies. In session 1.2 9 matches to the optimal policy are on average only 0.31 standard deviations apart from the rest, while the 5 matches to *hard* policy are more than 1 standard deviation different from others. Interestingly, here matches of the *hard* ad hoc policy are a subset of those who received the optimal policy. It may be that those who received the *hard* ad hoc policy that drive most of the distinctive power of optimal policy. In session 1.1, 7 recipients of the *hard* ad hoc policy are a subset of followers of optimal policy as well. In both session 1.1 and session 1.2, receiving a harder item at every step during a period of interest seems to be universally beneficial with respect to post-test result. In contrast, in session 7, where complying or not with the *easy* ad hoc policy distinguishes students far better than optimal policy. Here, an easier problem at each of the trials of interest is more beneficial. Note that in general the optimal policy is just aiming to maximize the expected student post test performance, and it may not outperform other policies in particular individual cases.

**Qualitative Assessment.** We also wished to further assess the resulting optimal instructional policies, using insight from the student model parameters. Table 3 shows the CFA model parameters that were fit using all 16 prob-

Table 2: Summary of student policy data.

| | Number of students | | | | | | | | | Cohen's D for post-test – pre-test | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Session | post-test | pre-test | follow optimal | follow hard | follow easy | follow optimal& hard | follow optimal & easy | Mean post-test score | Mean policy exp. score | follow optimal vs. others | follow hard vs. others | follow easy vs. others | follow optimal & hard vs. others | follow optimal & easy vs. others |
| 1.1 | 94 | 48 | 14 | 7 | 6 | 4 | 0 | 1.52 | 1.23 | 0.42 | 0.39 | -0.57 | 0.42 | N/A |
| 1.2 | 103 | 61 | 9 | 5 | 2 | 5 | 0 | 1.43 | 1.09 | 0.31 | 1.21 | -1.07† | 0.31 | N/A |
| 4 | 85 | 44 | 1 | 6 | 5 | 0 | 0 | 1.55 | 1.28 | -1.30† | -0.08 | -0.7 | N/A | N/A |
| 6.2 | 94 | 57 | 3 | 5 | 1 | 0 | 0 | 1.07 | 1.08 | -0.93† | -0.15 | 0.76† | N/A | N/A |
| 7 | 70 | 44 | 4 | 0 | 6 | 0 | 1 | 1.77 | 1.44 | 0.27 | N/A | 1.63 | N/A | 0.53† |

† Despite the values, bear in mind that the number of students following these policies is very low.

Table 3: Session 1, *Product* problems vs. *LCM* problems. User modeling parameters of recessed ($CFA_{1-6}$) and full ($CFA_{1-16}$) models with respective p-values

| Parameter | $CFA_{1-6}$ | $CFA_{1-16}$ |
|---|---|---|
| bias | -1.558(0.000***) | -0.824(0.000***) |
| $\beta_{Product}$ | 1.575(0.000***) | 1.143(0.000***) |
| $\gamma_{Product}$ | 0.109(0.482) | 0.124(0.020*) |
| $\rho_{Product}$ | 0.861(0.000***) | 0.219(0.002**) |
| $\gamma_{LCM}$ | 0.155(0.235) | 0.389(0.000***) |
| $\rho_{LCM}$ | 0.397(0.000***) | 0.080(0.019*) |
| $\gamma_{Product \to LCM}$ | 0.071(0.563) | -0.003(0.948) |
| $\rho_{Product \to LCM}$ | 0.554(0.000***) | 0.032(0.582) |
| $\gamma_{LCM \to Product}$ | -0.272(0.087.) | 0.094(0.036*) |
| $\rho_{LCM \to Product}$ | 0.209(0.067.) | 0.089(0.021*) |

lems in a session focused on teaching least common multiples. This model ($CFA_{1-16}$) has parameters that indicate learning from successes and failures for both *LCM* and *Product* problems. Transfer learning is significant and positive from a harder *LCM* to an easier *Product* problem, but the reverse direction (from *Product* to *LCM*) does not show significant transfer. This suggests that *LCM* problems help the student improve on both *LCM* and *Product* problems, but *Product* problems only produce improvement on *LCM* problems. Further this suggests that during tutoring it is likely to be more beneficial to provide *LCM* problems than *Product* problems.

For the LCM topic there were 14 out of 94 students that followed their respective optimal policies. The paths that these students took during trials 7 through 10 consisted of *LCM* problems only. This matches what we might expect given the $CFA_{1-16}$ model that demonstrates the particular transfer benefit of *LCM* problems. None of the paths of other 80 students were composed of solely *LCM* problems.

## 6. DISCUSSION

It is too preliminary to draw any definitive conclusions from this work because of the limitations of our dataset. From about 200-250 students in each session we had to select a subset that met our criteria of receiving different problem items on pre- and post- test trials. As a result the numbers shrunk to 70-100 students. Within this restricted set the student recipients of the 3 policies were very few.

There needs to be further work to better understand if simple policies are equally effective to the optimal policies. In this dataset we saw several instances of this. However, this could be due to fitting CFA models on a small data set covering only a few hundred students. It also could be because there was only a very small number of students where the problems selected matched any of the considered policies.

As part of the future work, we would like to repeat described experiments on several other datasets, potentially from different subject domains, where randomized data is available. Should the results turn out to continue support the preliminary evidence that optimized policies lead to better post-test performance, we would like to design an experiment using these policies to select skill practice for students.

## 7. ACKNOWLEDGEMENTS

This research was made possible with the assistance and funding of the U.S. Department of Education (IES-NCSER award #R305B070487), Carnegie Learning Inc., the Pittsburgh Science of Learning Center, DataShop team (NSF-SBE award #0354420) and Ronald Zdrojkowski.

## 8. REFERENCES

[1] H. Cen, K. Koedinger, and B. Junker. Comparing two IRT models for conjunctive skills. *ITS* 2008.

[2] M. Chi, K. Koedinger, G. Gordon, P. Jordan, and K. VanLehn. Instructional factors analysis: A cognitive model for multiple instructional interventions. *EDM* 2011.

[3] P. Pavlik, H. Cen, and K. Koedinger. Performance factors analysis - a new alternative to knowledge tracing. *AIED* 2009.

[4] P. Pavlik, M. Yudelson, and K. Koedinger. Using contextual factors analysis to explain transfer of least common multiple skills. *AIED* 2011.

[5] K. Tatsuoka. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20(4):345–354, 1983.

[6] W. van der Linden and R. Hambleton (Eds.), *Handbook of Modern Item Response Theory*. Springer, New York, NY, 1997.

# The real world significance of performance prediction

Zachary A. Pardos
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Rd.
Worcester, MA 01609, USA
zpardos@wpi.edu

Qing Yang Wang
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Rd.
Worcester, MA 01609, USA
wangqy@wpi.edu

Shubhendu Trivedi
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Rd.
Worcester, MA 01609, USA
shubhendu_trivedi@ieee.org

## ABSTRACT

In recent years, the educational data mining and user modeling communities have been aggressively introducing models for predicting student performance on external measures such as standardized tests as well as within-tutor performance. While these models have brought statistically reliable improvement to performance prediction, the real world significance of the differences in errors has been largely unexplored. In this paper we take a deeper look at what reported errors actually mean in the context of high stakes test score prediction as well as student mastery prediction. We report how differences in common error and accuracy metrics on prediction tasks translate to impact on students and depict how standard validation methods can lead to overestimated accuracies in these prediction tasks. Two years of student tutor use and corresponding student state test scores are used for the analysis of test prediction while a simulation study is conducted to investigate the correspondence between performance prediction error and latent knowledge prediction.

## Keywords

Evaluation, Cross-validation, Interpretation of error, Simulation

## 1. INTRODUCTION

An open question among EDM researchers and policy makers with an interest in EDM techniques is what impact the techniques reported on will have on students and what performance to expect under real world model training constraints. The majority of analytical papers presented in the literature using educational datasets use n-fold cross-validation. This has become an expected standard and a justifiable one which offers clear statistical reliability benefits over a single test and train hold out strategy. However, as an applied field it is important to take a step back from the manipulation of datasets and consider what factors may impact the expected performance in a real world deployment of a method. Often the culprit of inflated cross-validated accuracy is the disregard for time constraints in temporal data. Because this type of data is predominant in the field due to the temporal nature of studying learning, it is especially important to keep violations of time in mind in the evaluation and reporting of our models.

Data leakage [1] is the more general term for using information during training or prediction that should not legitimately be available. This kind of leakage of data from the future has been prevalent in many data mining competitions including the 2010 KDD Cup on educational data [2]. In that competition, for example, a student's answers from Unit 2 could be used to predict her responses to questions of a related skill in Unit 1. While the models which were designed to predict that type of test set may very well also push the state of the art in real world prediction scenarios, the prediction accuracies reported in that competition do not reflect real world performance expectation. Furthermore, the relative rankings of algorithms in the competition may vary when future information is not available. We investigate the effect of leakage on the task of predicting end of year state test scores in section 2.

Removing leakage from evaluation adds confidence in replicating the reported error in real world deployment, however; of equal significance to deployment considerations is the real world meaning of the error and its implications for students. Recent work on ensemble learning with educational data [3] chronicles the various models introduced in the past years which track student knowledge. A common practice among these papers has been to compare the error of a newly introduced method to that of a longer established method. Generally, the merit of the new method is compelling if it demonstrates a statistically reliable improvement in error over the established method. With larger educational datasets becoming widely available, such as the 20M row 2010 KDD Cup dataset[1] [2] or the 1M row ASSISTments Platform dataset[2], finding statistical differences in models can be achieved even with prediction error differences among models only discernible at the third or fourth decimal. This raises the question of whether or not statistical tests are a useful yard stick when large datasets are being analyzed and more importantly it raises the question of what errors and various magnitudes of differences in errors actually mean in terms of their impact on students. The most practical application of these models, and a reason for their high relevancy in the literature, is to predict when a student has attained mastery of a particular skill. Improved accuracy of these knowledge tracing models is appealing because it presumes that the prediction of mastery will also be more accurate and thus reduce the amount of over and under practice on skills, a time saving benefit that teachers greatly appreciate. In section 3 we run a simulation study to investigate the meaning of errors in terms of knowledge assessment. In the simulation study both student knowledge and response data is generated from a standard model of learning. We evaluate the generated response data with several models to evaluate the correspondence between performance prediction metrics and accurately inferring when mastery was attained.

Best practices for calculating statistically reliable difference between predictions is an open question, however, a frequent approach is to calculate a paired t-test of prediction squared errors [2] or a Wilcoxn signed rank sum test on per student Area Under the Curve (AUC) also referred to as A' [3].

[1] http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

[2] http://teacherwiki.assistment.org/wiki/Assistments_2009-2010_Full_Dataset

## 2. PREDICTION OF STATE TEST SCORES

In this section we evaluate the effect of leakage in predicting state test scores and also provide an analysis of the impact of error differences on student test score prediction.

### 2.1 Dataset

We used two datasets [4] collected from the 2004-2005 and 2005-2006 school years usage of the ASSISTments Platform among 8th grade math students in four Massachusetts high schools. The datasets had 627 and 760 students respectively. Both datasets were organized with one row per student and six features summarizing each student's usage in the system for that year. The per student features were: overall percent correct, number of problems answered, percent correct on scaffold questions, average time spent per problem, average attempts per problem, and average number of hints requested per problem. The seventh feature is the student's end of year math state test score, which is the target being predicted. The state test is the Massachusetts Comprehensive Assessment System (MCAS). The minimum raw score for the test is 0 and the maximum score is 54. The raw score is scaled to a score between 200 and 280. The scaled score contains four ranges that correspond to the following proficiency categories shown in Table 1.

| Score Range | Category |
|---|---|
| 200-218 | Failing |
| 220-238 | Needs Improvement |
| 240-258 | Proficient |
| 260-280 | Advanced |

**Table I.** Proficiency categories for the MCAS test

While the scaled score ranges always map to the same categories, the raw score mapping to scaled score changes yearly and is only computed after all tests are received and evaluated by the state. This presents an additional challenge for category prediction; however, scaling is just one of many sources of change between the two years' data. Changes in the tutor as well as changes in student instruction outside the tutor also contribute variance and are a part of why a two year train/test procedure might be more difficult to predict than a one year cross-validated.

The MCAS test is a high stakes test because of the significance of scoring in the Failing or Advanced categories. Failing category students cannot graduate high school, regardless of their class grade, while students who score in Advanced receive an automatic state college scholarship. For this reason, interested parties want to know a prediction of the student's category, not just raw score.

### 2.2 Methodology

Two prediction algorithms were used and two hold out strategies. Multiple algorithms were chosen not for the sake of comparison but rather to see if the relative performance of the algorithms changes between hold out strategies. An algorithm that does not fit the cross-validated set very well may capture the appropriate level of generality to be better in the train/test scenario. The two prediction algorithms chosen were linear regression, used in prior work with this dataset [4] and Random Forests [6], a highly affective algorithm from the machine learning community. We also include a K-means clustering technique that claimed to improve prediction accuracy of algorithms on this same dataset [5]. This K-means enhancement is an ensemble technique [3] and we include it to see if it underperforms in the train/test hold out.

Two hold out strategies were used, one to demonstrate a typical 5-fold cross-validation hold out which contains future information leakage and the other strategy uses the previous year's data to train the algorithm and uses the next year's data to test on. For the cross-validation, the 2005-2006 data was used. The second hold out strategy represent a realistic scenario where only historic data is able to be used to train a model whereas the cross-validated hold out allows information about test outcomes and scaling which the algorithms should not legitimately have access to.

The actual distribution of students in the different categories is presented as well as the predicted distribution according to the various algorithms paired with the two hold out strategies. The error metrics used are Mean Absolute Difference (MAD) and Root Mean Squared Error (RMSE). The formula for RMSE is:

$\sqrt{\frac{\sum_{i=1}^{n}(predicted_i - actual_i)^2}{n}}$ where $n$ is the number of students.

### 2.3 Results

The results presented in Tables II and III are the algorithm errors in predicting the raw test scores. Scaled score false positive and negative evaluation is reported in Tables VI and V.

| Algorithm | RMSE | MAD |
|---|---|---|
| Linreg + K-means | 9.193 | 7.240 |
| Linreg | 9.262 | 7.358 |
| RF + K-means | 9.399 | 7.463 |
| RF | 9.420 | 7.540 |

**Table II.** MCAS prediction error using 5-fold cross-validation on the 2005-2006 data

Linear regression with bagged K-means resulted in the most accurate prediction of test scores according to RMSE and MAD error metrics. The more complex prediction technique of random forests did not fare as well with regular linear regression beating random forests with bagged K-means and random forests alone. The RMSE difference between the best and worse algorithm was 0.227, or 2.4% worse than the best score.

We now compare to the other hold out strategy where the same 2005-2006 test scores are being predicted except using data from 2004-2005 to train. Prediction results of this second hold out strategy are shown in Table III.

| Algorithm | RMSE | MAD |
|---|---|---|
| Linreg + K-means | 9.748 | 7.957 |
| Linreg | 9.817 | 8.044 |
| RF + K-means | 9.941 | 8.204 |
| RF | 10.106 | 8.337 |

**Table III.** MCAS prediction error of 2005-2006 test scores using 2004-2005 data.

Table III shows that the relative rankings of the algorithms have not changed using this hold out strategy but the overall errors have increased. The RMSE difference between the best score using cross-validation versus using the previous year's data is 0.555, or 6% worse than the better score. This difference is more than twice the difference between the best and worse algorithms in Table II. What does this level of difference mean to actual student score prediction? To investigate this we look deeper at the predicted score category compared to the actual category of the

two best algorithms using each hold out strategy. For the train/test hold out strategy we used the '04-'05 scaling to transform the '05-'06 raw predictions to categorical predictions.

| | real | pred. | false pos. | false neg. | sensitivity |
|---|---|---|---|---|---|
| **Adv.** | 0.083 | 0.016 | 0.003 (2) | 0.841 (53) | 0.159 |
| **Prof.** | 0.176 | 0.140 | 0.099 (62) | 0.672 (90) | 0.328 |
| **Need.** | 0.364 | 0.530 | 0.444 (215) | 0.321 (89) | 0.679 |
| **Fail.** | 0.377 | 0.315 | 0.171 (81) | 0.445 (128) | 0.554 |

**Table IV.** Statistics for Linear regression + bagged K-means prediction of the cross-validated 2005-2006 data

Table IV shows the real percentage of students that fall in to the four proficiency categories as well as the predicted percentages according to the prediction algorithm. False positives, false negatives and sensitivity are also shown. Sensitivity is the probability that students who belong to that category will be properly placed into that category. We can observe that not many students scored in the advanced category and that the majority of the distribution (74.1%) lies in the Needs Improvement and Failing categories. Two students were placed in advanced that did not belong there and 53 students were not placed in that category that belonged there. For failing, 81 students were placed there that did not belong there and 128 were not placed there who belonged there.

| | real | pred. | false pos. | false neg. | sensitivity |
|---|---|---|---|---|---|
| **Adv.** | 0.083 | 0.000 | 0.000 (0) | 1.000 (63) | 0.000 |
| **Prof.** | 0.176 | 0.152 | 0.1132 (71) | 0.664 (89) | 0.336 |
| **Need.** | 0.364 | 0.654 | 0.654 (286) | 0.235 (65) | 0.765 |
| **Fail.** | 0.377 | 0.193 | 0.089 (42) | 0.634 (182) | 0.366 |

**Table V.** Statistics for Linear regression + bagged K-means prediction of 2005-2006 data using 2004-2005 data to train

In Table V we can see a different distribution that places the bulk of the classification into the Needs Improvement category. Looking again at the most important categories, the advanced had no false positives but had 63 (100%) students placed outside of advanced that should have been in advanced. For failing, 42 students were improperly classified into that category while 182 were improperly left out of the category.

The cross-validated hold out misclassifies more students into Failing while the non cross-validated hold out fails to correctly classify more students as Failing. Both hold out strategies fail to classify all or most of the advance students as advanced.

This analysis demonstrates the areas of improvement for this test score prediction task, particularly in correctly identifying Advanced students. The hold out analysis also shows that while the previous year training strategy resulted in 6% worse error, it is still performing reasonably well compared to the cross-validated result in important category classification areas according to the statistical analysis. This more in-depth analysis gives us confidence that deployment of this prediction method in a real world setting would result in raw test score predictions of within 12% of actual (8/54). Misclassification of Advanced and Failing students is an aspect that needs improvement on the algorithm end, perhaps with ensemble techniques or the addition of more features engineered from the logged data.

# 3. INFERRING STUDENT KNOWLEDGE

In this section we conduct a simulation study to observe the correspondence between performance prediction error and knowledge inference error. In particular, at which opportunity does the model infer knowledge has been attained compared with the opportunity at which the simulated student attained knowledge in the synthesized data. This correspondence is compared with the prediction error of each model. The significance of performance prediction is looked at from a different angle than in the previous section. Instead of measuring the effect of leakage on prediction, we look at how performance prediction corresponds to a different objective, that of inferring student knowledge. This type of inference of knowledge is used in the Cognitive Tutors [8]. Reported performance prediction improvements often come with the presumption that knowledge inference accuracy is also improved.

## 3.1 Dataset

For this dataset we synthesized data for 500 simulated students answering 50 questions each of the same skill. The simulation generated 50 responses per student in addition to 50 knowledge states per student. Student responses are either 0, representing an incorrect answer, or 1, representing a correct answer. Student knowledge states are also 0 or 1 corresponding to the skill being known or not known.

## 3.2 Methodology

The standard Bayesian Knowledge Tracing [7] model was used to simulate data. This is a Hidden Markov Model of learning where a student is either in the learned or unlearned state and evidence of their past response history can be used as evidence to infer the probability of their current knowledge state as well as the probability of a correct answer on the next problem opportunity. The model has four parameters: prior, learn, guess and slip and these parameters were fixed to values of 0.30, 0.09, 0.14 and, 0.09 respectively for the generation of the data.

A 5-fold student level cross-validation was run using six different knowledge tracing models to attempt to recover the parameters and predict simulated student response and also infer the probability of knowledge at each opportunity. The six models included: 1) the ground truth model (GT) using the real generating parameters 2) a model that let Expectation Maximization (EM) iterate until convergence 3-6) these models kept three parameters at their ground truth values and increased the fourth by 0.20. For example, model "gt_guess" has the guess parameter set to 0.34 instead of 0.14 while all other parameters remain at ground truth level. These models were included so we may observe the sensitivity of the various parameters on performance and knowledge prediction. RMSE was again used to evaluate results as this has been a popular metric to evaluate within-tutor prediction and was the metric used to score results in the 2010 KDD Cup challenge [2]. AUC was used in place of MAD as AUC has also been popular in the user modeling literature to score prediction accuracy. AUC can only be used with binary prediction classes and so it was not applicable to the MCAS scoring. AUC is an accuracy metric with a 0.50 score being no better than chance and a score of 1 being a perfect prediction. Statistics comparing the correspondence between the time that simulated student knew the skill and the time that the inferred probability of knowledge was 0.95 or above were also calculated. The threshold of 0.95 is common in Cognitive tutors [8] for determining that a student has mastered a skill and allowing them to move on in the curriculum.

## 3.3 Results

| model | RMSE | AUC |
|---|---|---|
| EM | 0.4273 | 0.7260 |
| GT | 0.4296 | 0.7191 |
| gt_prior | 0.4307 | 0.7154 |
| gt_guess | 0.4367 | 0.7998 |
| gt_slip | 0.4373 | 0.7241 |
| gt_learn | 0.4773 | 0.6480 |

**Table VI.** Cross-validated simulation performance prediction results for the eight models.

Table VI shows that the best model according to RMSE was the EM model. The gt_guess model was best according to AUC. It is somewhat surprising that the ground truth model, although close in RMSE, was not the best. The EM model converged to within 0.01 of GT parameter values, so the slightly improved accuracy may be due to chance that this particular simulated population skewed towards the EM converged parameters. AUC is a rank order estimation of accuracy and thus, so long as predictions correlate with responses, the predictions can be poor and still attain high AUC. Nevertheless, this result is surprising. The worse model, according to both AUC and RMSE, was the model which increased the learn rate parameter by 0.20. This suggests that learn rate is a sensitive parameter to prediction error and a potentially worthwhile area to focus on for student prediction improvement.

| | Median un/over predicted | Mean absolute difference | students over practiced | students under practiced |
|---|---|---|---|---|
| gt_learn | 1 | 2.37 | 409 | 56 |
| GT | 2 | 2.53 | 469 | 21 |
| gt_prior | 2 | 2.53 | 469 | 21 |
| gt_slip | 2 | 2.64 | 473 | 17 |
| EM | 2 | 2.68 | 475 | 18 |
| gt_guess | 4 | 4.81 | 494 | 0 |

**Table VII.** Under and over practice amounts on average caused by model inference in students' knowledge using a mastery threshold of 0.95 probability.

Table VII shows how each model performed at inferring when a student has mastered the skill. The median un/over practice column shows the median number of over or under practice opportunities. Average of absolutes column calculates the average absolute under/over prediction which takes the absolute value of the residual between inferred mastery opportunity and actual mastery opportunity. The lower this value, the better the model did at inferring exactly when a student learns and not letting them over or under practice the skill. As we can see by the "number over practiced" column, the vast number of simulated students are inferred to learn the skill after they have actually learned it. The worse over predictions was by the gt_guess model caused decreased confidence in knowledge when observing positive performance which further exacerbated the under prediction bias.

# 4. DISCUSSION

We have investigated the significance of performance prediction in the context of test score prediction and within-tutor knowledge inference. We have raised the issue of leakage in prediction evaluation and its role in cross-validation accuracy inflation. The result of leakage was a 6% increase in error from the best cross-validated model to the best model trained on the previous year's data. A 6% increase is reasonable for training on a separate cohort of students. An additional analysis of the results using a confusion matrix revealed a decrease in prediction of proficiencies at the extremes, and a tendency to predict more towards the average proficiency category with the previous year hold out.

Our simulation study revealed a clear bias towards knowledge under prediction among the knowledge tracing models. The inflated learning rate model, gt_learn, worked to offset some of this bias, reducing the median over prediction from 2 to 1 opportunity, which provided a better knowledge inference estimate but also resulted in the worst performance prediction score. This discord underscores the motivation behind studying the real impact of performance prediction on the intended objectives, although this magnitude of disparity warrants further investigation.

# 5. ACKNOWLEDGEMENTS

This research was supported by the National Science foundation via grant "Graduates in K-12 Education" (GK-12) Fellowship, award number DGE0742503. We thank Feng & Heffernan for the datasets and the Worcester Public Schools for their ongoing partnership and for making test scores available to us. We also acknowledge the many additional funders of ASSISTments Platform found here: http://www.webcitation.org/5ym157Yfr

# 6. REFERENCES

[1] S. Kaufman, S. Rosset, and C. Perlich. Leakage in data mining: formulation, detection, and avoidance. In Proceedings of the 2011 conference on Knowledge Discovery in Data Mining, San Diego. pp. 556-563.

[2] Pardos, Z.A., Heffernan, N. T.: Using HMMs and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. To appear in the Journal of Machine Learning Research W & CP, In Press

[3] Gowda, S., Baker, R.S.J.d., Pardos, Z., Heffernan, N. (2011) The Sum is Greater than the Parts: Ensembling Student Knowledge Models in ASSISTments. Proceedings of the KDD 2011 Workshop on KDD in Educational Data.

[4] Feng, M., Heffernan, N.T., & Koedinger, K.R. (2009). Addressing the assessment challenge in an Intelligent Tutoring System that tutors as it assesses. The Journal of User Modeling and User-Adapted Interaction. Vol 19: p243-266.

[5] Trivedi, S., Pardos, Z. & Heffernan, N. (2011) Clustering Students to Generate an Ensemble to Improve Standard Test Score Predictions In Biswas et al (Eds) Proceedings of the Artificial Intelligence in Education Conference 2011. Springer. LNAI 6738, Pages. 328–336.

[6] L. Breiman. Random forests. Machine Learning, 45(1):5-32, 2001.

[7] CORBETT, A. T., & ANDERSON, J. R. 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, 4, 253-278.

[8] CORBETT, A. T., KOEDINGER, K., & HADLEY, W. S. 2001. Cognitive tutors: From the research classroom to all classrooms. In P. S. Goodman (Ed.), Technology enhanced learning: Opportunities for change. (pp. 235-263). Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers.

# The Rise of the Super Experiment

John C. Stamper
Carnegie Mellon University
Pittsburgh, PA
john@stamper.org

Derek Lomas
Carnegie Mellon University
Pittsburgh, PA
dereklomas@gmail.com

Dixie Ching
New York University
New York, NY
dixie@nyu.edu

Steve Ritter
Carnegie Learning, Inc.
Pittsburgh, PA
sritter@carnegielearning.com

Kenneth R. Koedinger
Carnegie Mellon University
Pittsburgh, PA
koedinger@cmu.edu

Jonathan Steinhart
Vienna, Austria
jsteinhart@gmail.com

## ABSTRACT

Traditional experimental paradigms have focused on executing experiments in a lab setting and eventually moving successful findings to larger experiments in the field. However, data from field experiments can also be used to inform new lab experiments. Now, with the advent of large student populations using internet-based learning software, online experiments can serve as a third setting for experimental data collection. In this paper, we introduce the Super Experiment Framework (SEF), which describes how internet-scale experiments can inform and be informed by classroom and lab experiments. We apply the framework to a research project implementing learning games for mathematics that is collecting hundreds of thousands of data trials weekly. We show that the framework allows findings from the lab-scale, classroom-scale and internet-scale experiments to inform each other in a rapid complementary feedback loop.

## Keywords

eScience, experiment design, internet scale

## 1. INTRODUCTION

Web-based software is creating an explosive growth in the use of randomized controlled experiments in education, due to the relative ease with which users can be randomly assigned to different experimental conditions. Scientists are beginning to recognize the coming data surge and developing new ways of analyzing data at "internet scale." The vastly increased scale of subject populations online can produce a categorically different mode of experimentation in education. For this reason, we propose a new experimental framework that takes advantage of rapid internet-scale experimentation, while retaining the control of lab-scale and classroom-scale experiments.

Randomized controlled trials are regularly used to drive design decisions on the internet. In its simplest form, A/B testing is a form of experimentation where one of two advertisements are randomly delivered to each incoming site visitor. This allows advertisers to determine which advertisement results in improved outcomes (such as a greater click-through rate) [3]. Multiple tools exist to support website optimization, including the free Google Site Optimizer that supports both A/B tests and multi-variable testing. Recently, free-to-play online game companies, such as Zynga, have made use of large-scale optimization experiments with their large number of online players. By randomly assigning players to hundreds of different game design configurations, they can optimize the game design to maximize the conversion of players to paying customers [7].

## 2. INTERNET SCALE RESEARCH IN EDUCATION

Internet-scale research introduces new potential methods in Educational Research. For instance, optimization experiments like Response Surface Methods, are a common applied research method for improving industrial process outcomes. These experimental designs showed early promise for improving educational outcomes [5], but because the designs would have required many hundreds of students, they were expensive and impractical. Internet-scale research can now support these optimization experiments, along with these other experimental advantages:

*Increased number of conditions*. With tens of thousands of "user-subjects," internet-scale research studies present the opportunity for researchers to run dozens—even hundreds—of different experimental conditions simultaneously. This easily contrasts with lab or field-scale studies, where available resources and subject pools typically constrain experimental designs to fewer than 8 experimental conditions. Furthermore, with fewer conditions, experiments can be conducted within days, rather than months.

*Ability to measure "true" task engagement.* Internet-scale research is also uniquely suited for measuring task engagement. Because the researcher typically lacks control over participants (they can quit far more easily than in lab or classroom experiments), the internet is an ideal setting for investigating user motivation. If players assigned to condition A play significantly longer than players in condition B (i.e., were engaged in the task for longer), then condition A can be said to be more engaging than condition B. The ability to measure and compare engagement makes it possible to measure how different design elements and configurations affect player engagement.

*Increase in external validity.* A third advantage of internet-scale research is the high external validity—experiments are conducted with actual "real-world" users. While the lack of control over subjects can result in noisy data, this noise is useful for preventing over the over-fitting of predictive models that constructed for use "in the wild."

*Greater access to all users.* A fourth advantage of internet-scale research is the fact that informed consent is not required if the users are anonymous. Even with educational exemptions to informed consent, parental opt-out forms can still pose a barrier to

many field-based educational studies. While researchers could potentially make use of informed consent (and thus obtain non-anonymous data), anonymous data collection is likely to remain a characteristic of most large internet-scale research.

Of course, the lack of information about participants is also a key drawback of internet-scale research. Broadly speaking, internet scale studies cannot collect rich information about participants. Therefore, these studies are unlikely to be suitable when research questions require demographic data, detailed pre/post tests, participant observation, talk-aloud protocols, or any kind of psychophysiological measure. Finally, the lack of participant control means that internet scale studies may not be appropriate if repeated participation over time is required.

Given these drawbacks, it is clear that traditional lab based experiments and structured field trials still provide valuable data that internet scale experiments cannot. However, there is much to be gained from internet scale studies. The Super Experiment Framework (SEF) seeks to illustrate how different scales of experimentation can productively inform one another. The SEF framework, seen in Figure 1, is split into three general experimental parts that are roughly delineated by scale. Lab-Scale experiments are smaller highly controlled studies that take place in a lab or single classroom, generally not exceeding 50 participants. School-Scale experiments are formal experiments that take place in multiple classrooms or schools consisting of hundreds to thousands of participants. Internet-Scale experiments are informally delivered online to thousands to millions if participants.

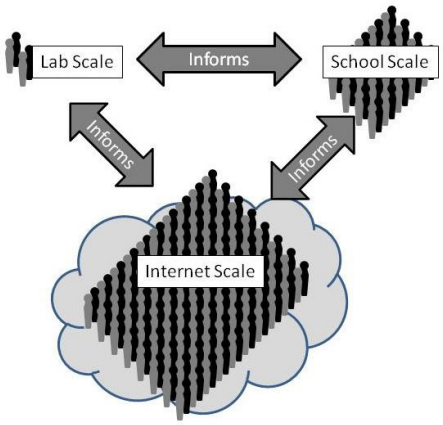



**Figure 1. The Super Experiment Framework showing how each of the component scales informs the others.**

In the SEF framework, each component provides an experimental level that can be used to answer specific questions that might be difficult or impossible to answer using one of the other components. Further, the various components can be used to expand or validate findings of the other components. A feedback loop can also be used with the framework where internet scale experiments can identify areas of focus for lab scale experiments, which can then be validated in school scale experiments. An overview of each of the SEF components can be seen in Table 1.

School scale and lab scale experiments typically recruit subjects and then randomly assign them to different experimental conditions as part of a single experiment. However, internet-scale research creates situations where multiple experiments are randomly drawing from the same pool of subjects. Just as a single experiment contains multiple experimental conditions, the SEF contains multiple experiments. Because the different experiments are derived from the same pool of random assignment, experimental conditions that are not part of the same experiment may still be compared to one another, if desirable. While there may be few immediate benefits of this comparison, the super experiment is a unique characteristic of internet-scale research. Therefore, the use of the term "super experiment" in the super experiment framework simply refers to the broad network of information flow between different scales of experimentation, from the lab scale, to the school scale and to the internet scale.

| Type | Benefits | Drawbacks |
|---|---|---|
| Lab Scale (1-75) | Rich user data, Formal, Controlled CTA, Talk alouds, Psycho-physiological studies | Effect Size, Replication, Scalability, Experimenter effects, Threats to external validity |
| School Scale (25-10,000) | Formal, Controlled, Validation, Good randomization, Surveys, Enforced participation | Expensive, Difficult to replicate, Threats to external validity |
| Internet Scale (10,000-millions) | Informal, Large data collection, Rapid, High external validity, Decreased Type II error rate, High power | Anonymity, High attrition, Data overload, Threats to internal validity |

**Table 1. Components of the Super Experiment Framework**

## 3. IMPLEMENTATION EXAMPLE

The need for the SEF framework was initiated through our work in creating online games for learning. The number of potential experiments was large and the opportunity to field the games at each of the scales identified in the SEF framework provided the need to build a feedback loop to execute many experiments at internet scale in order to narrow down the potential experiments to test at the more controlled school scale. "Battleship Numberline" (BSNL), an online educational game, benefits from the super experiment framework.

Designed to improve number sense among elementary and middle school students, BSNL provides practice estimating numbers on a number line within four content domains: whole numbers, fractions, decimals and measurement [4]. The game narrative involves defending Numbaland Island from invading robot pirates by firing projectiles at their ships and submarines. BSNL involves two basic modes: naming numbers and placing numbers. In the naming condition, players type a number that corresponds to the location of an enemy ship that is positioned on a number line between two marked endpoints. In the placement mode, the player is given the numeric location of a hidden submarine (e.g., "Submarine spotted at 1/3") and needs to click on the location that they believe corresponds to the number. After the player has typed a number or clicked on the number line, a projectile drops vertically from the top of the screen to the designated location on the number line. Animation and text-based feedback communicates the player's accuracy after every round.

A primary goal of our research has been to understand how different game design factors affect player learning and engagement. In order to systematically investigate these factors, we implement these design factors as flexible xml-based

parameters that can be determined at the game runtime. We are then able to create online experiments that randomly assign new players to a set of different game sequences.

During gameplay, BSNL generates an online data log of the task context (the above xml parameters) along with data describing the player's performance on each opportunity. On each item, we log the player's reaction time, their accuracy, and a binary field indicating whether the player was successful or not. Logs are then imported into the PSLC Datashop [2], which allows for the secondary analysis of player performance and learning. The hit rate measure is essential for enabling Datashop to plot learning curves of error rate over time. By labeling different items in the game with different knowledge components (e.g., reducible fractions, unit fractions, etc), we can plot learning curves for each knowledge component. Learning curves can also be described based on fluency [1], where we plot the reduction of reaction time over opportunities played. In addition to these measures of learning and performance, we investigate player engagement through two measures: the total number of items played and the total amount of time spent playing. These two metrics correspond with our construct of intrinsic motivation or player engagement.

The number of potential parameter settings in BSNL makes it a great tool to answer many research questions, but at the same time the number of possible settings make it difficult to decide on what settings to in traditional lab or school settings. For this reason, it is a perfect candidate for use in the SEF. Next, we show how the results of different types of experiments at one scale inform new experiments on a different scale.

*Lab Scale informing School Scale*. The use of a lab experiment to inform a field trial at a school is one of the most common types of experimental design. It is still an important part of the SEF. We performed a lab scale experiment, which is now being validated at the school scale. This experiment was conducted at a small Catholic liberal arts University. Although the college is co-educational, its focus is on women's education, and 89% of the participants were women. Participants were 18 students in an eight-week first-year seminar course, which met once per week. Students chose for this seminar period to focus on mathematics games. Over 5 weeks, we administered a short (typically one minute) paper-and-pencil pretest, asked students to play a specific fluency game for approximately one-half hour and then gave a posttest which was identical in content to the pretest. In all but the first week, the pretest was preceded by a delayed post-test, which was a repeat of the posttest from the previous week's materials.

In four of the five experiments significant improvement was shown on a delayed post-test, and three of the five showed immediate results. Effect sizes were also quite large, ranging from 0.4 to 2.4, indicating that these results are not only significant but substantial. Prior to the first experiment, students were given a survey about their confidence in mathematics (containing questions like "I am sure that I can learn math.") and about text anxiety (containing questions like "I am so nervous during a test that I cannot remember facts that I have learned"). The two scales were mixed in a 16-item form. Students were asked to rate each statement from 1 ("strongly disagree") to 5 ("strongly agree"). Student confidence increased significantly, $t(14)=-3.2$, $p<.01$, $d=0.4$, but there was no change in test anxiety, $t(14)=-3.1$, n.s.

Due to the success of this lab scale experiment, a similar school scale experiment is now being conducted in multiple college classrooms over an entire semester. Unlike the lab scale, the researchers are not present in these classrooms, but we expect to see similar results.

*School Scale informing Internet Scale.* BSNL was designed based on an existing body of literature that investigated number line estimation in the laboratory [6]. The game was playtested with 8 elementary school students, to refine usability issues in the design. Following this, a school scale study was conducted with 119 students in grades 4-6. Students showed significant improvement in hit rate form the first to second opportunity (see Figure 2), and students demonstrated significant improvements in the estimation of fractions on a number line after 20 minutes of gameplay. Moreover, 82% of players (74% females, 92% males) reported that they wanted to play the game again [4]. The data from these classroom studies was imported into the PSLC Datashop to test various knowledge component (KC) models. We identified a KC model based on the various regions of the number line. This knowledge component model was then used to produce a Bayesian Knowledge Tracing adaptive sequencing algorithm. This algorithm was then tested online in comparison with a randomly sequenced level. Preliminary results suggest that the BKT adaptive sequence did not result in significantly greater player engagement than the random sequence.

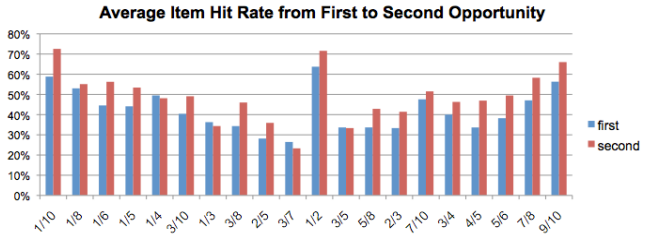



**Figure 2. Illustrates the average improvement from the first opportunity to the second opportunity, by item presented. The clear patterns of difficulty are used to generate knowledge component models in Datashop.**

*Internet Scale informing School Scale or Lab Scale*. Internet-scale experiments can be useful for documenting the difficulty of different task configurations. This is useful in the field of EDM, as it allows for the generation of knowledge component models. Different tasks are said to require different knowledge components if and only if the tasks result in different performance rates or learning curves. Therefore, by assessing the difficulty of instances over a broad task design space, we can understand how the task design space maps to various KC models.

For example, Rittle-Johnson, Siegler and Alibali found that tickmarks supported the estimation of decimals on a number line [6]. In order to replicate this work and extend it, we randomly assigned online players to 6 different conditions in both the decimal and whole number domain. Players either encountered tickmarks dividing the number line into tenths, fourths, thirds, halves (midpoint), or no tickmarks at all. Finally, an additional two conditions looked at the interaction of an adaptive sequencing algorithm with tickmarks at the midpoint. An overview of the experiments and conditions can be seen in Table 2. Over 80,000 internet users participated in the experiment.

An experiment with this many conditions would be difficult to replicate in a lab or classroom. This broad investigation of the effects of guides enabled us to observe two unusual outcomes. First, there was an apparent interaction effect between our adaptive sequencing condition (termed "ITS") and the midpoint

guides. Neither Second, the 10th guides apparently increased player engagement in the decimal condition, but decreased engagement in the whole number condition. These insights have led us to execute similar lab scale experiments to replicate and better understand these specific results.

| Experiment Name | Conditions | Players |
|---|---|---|
| Adaptive Sequencing | 15 | 19,856 |
| Difficulty Sequencing | 6 | 6,302 |
| Difficulty Comparison | 6 | 6,234 |
| Expanded fraction set | 4 | 5,596 |
| Guides Engagement | 10 | 11,386 |
| Guides Learning | 20 | 22,441 |
| Measurement Study | 3 | 10,014 |
| **Total** | **64** | **81,829** |

**Table 2. List of experiments running concurrently with a total of 64 conditions.**

# 4. CONCLUSIONS AND FUTURE WORK

Technology is forever changing the way we conduct experiments. The traditional paradigm is no longer the best way to do things. Data is coming in faster, larger, and more fine grained. Instead of focusing eScience efforts in just analyzing we have created a framework to exploit internet scale experiments, while still creating valid findings in real classrooms.

The main contribution of this work is the development of the Super Experiment Framework which incorporates a feedback loop allowing for experiments of different scales to inform each other. This has become possible, and even necessary, with the use of the internet to collect a large amount of experimental data. Internet scale allows for optimization experiments that would be too expensive to do at field level. This is truly applied educational research that, as we have shown, provides insights that can inform more controlled lab or school scale experiments. We also explained our initial implementation of the SEF with a large project with broad scope and many interesting research questions. Traditional "one-way street" experiments of lab to school are slow to findings and outdated. Our work shows how utilizing all three scales of experiments leads to rapid findings that can lead to real implementable insights efficiently.

Making the framework possible is the accessibility of internet scale experiments. The key barrier to internet scale educational research is attracting large numbers of users. Research projects rarely invest in high-quality software design and usability, which is usually necessary to achieve widespread adoption. However, once this quality is developed, large numbers of users can be reached through collaborations with one of many internet portals that seek to aggregate educational content (e.g., Brainpop.com).

Another challenge is instrumenting software for generating data logs that measure player performance, learning and engagement. Log files should capture not only correctness information, but the amount of time that players spend on an activity, as well as the number of opportunities attempted to make these measures.

A third challenge is the configuration of the software to allow for experimental designs. This involves the abstraction of design variables in the software's design space, such that different instances of the software can be created quickly. For instance, we use xml to define game levels at run-time. These configurations can then serve as different experimental conditions that can be randomly deployed to online users.

Finally, one unusual new challenge in internet scale research is the efficiency of subject-pool utilization. While lab or school scale researchers expend significant effort to recruit a sufficient number of subjects in order to achieve statistical significance, internet scale researchers increasingly face the challenge of making use of tens of thousands of subjects in an efficient manner. Certain types of experimentation may result in inconsistent user experiences that reduce overall participation.

Some challenges will be particular to individual experiments. For instance, in our online experiments we observe strong seasonal effects of weekends and school holidays, where the number of players is greatly reduced. This suggests that certain experimental comparisons should be sensitive to the time period of the study, not merely the number of subjects.

Many of these challenges can be mitigated by validating the results of internet scale experiments with controlled classroom experiments. As shown in the experiment section, we are continuing to run a number of experiments of scales based on findings of different scales. This feedback loop will continue in the future as we strive to optimize the games to maximize learning. We believe this framework will rapidly lead to significant discoveries that are replicable at each of the scales.

# 5. ACKNOWLEDGMENTS

We would like to thank the Pittsburgh Science of Learning Center, the DataShop staff, the Next Generation Learning Challenge, Carlow University, and Pellisippi State University for supporting this research.

# 6. REFERENCES

[1] Baker, R., Habgood, M., Ainsworth, S., & Corbett, A. Modeling the acquisition of fluent skill in educational action games. User Modeling, 4511, (2007), 17-26.

[2] Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. (2011) A Data Repository for the EDM commuity: The PSLC DataShop. Handbook of Educational Data Mining. CRC Press

[3] Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. Controlled experiments on the web: survey and practical guide. Data Mining and Knowledge Discovery, 18(1), (2008), 140-181.

[4] Lomas D., Ching D., Stampfer, E., Sandoval, M., Koedinger, K. Battleship Numberline: A Digital Game for Improving Estimation Accuracy on Fraction Number Lines. *Conference of the American Education Research Association (AERA)* (2012).

[5] Meyer, Donald L. Response Surface Methodology in Education and Psychology, Journal of Experimental Education, 31, 4, (1963), 329-336.

[6] Rittle-Johnson, B., Siegler, R. S., and Alibali, M. W. Developing conceptual understanding and procedural skill in mathematics: An iterative process, *Journal of Educational Psychology*, 93, (2001), 346-362.

[7] Sheffield, B. GDC Canada: Bill Mooney Outlines Zynga's Methodology For Success, Gamasutra, May 6, 2010. Retrieved 2/10/12:http://gamasutra.com/view/news/28434/

# Using Student Modeling to Estimate Student Knowledge Retention

Yutao Wang
Worcester Polytechnic Institute
100 Institute RD.
Worcester, MA, USA
yutaowang@wpi.edu

Joseph E. Beck
Worcester Polytechnic Institute
100 Institute RD.
Worcester, MA, USA
josephbeck@wpi.edu

## ABSTRACT

The goal of predicting student behavior on the immediate next action has been investigated by researchers for many years. However, a fair question is whether this research question is worth all of the attention it has received. This paper investigates predicting student performance after a delay of 5 to 10 days, to determine whether, and when, the student will *retain* the material seen. Although this change in focus sounds minor, two aspects make it interesting. First, the factors influencing retention are different than those influencing short-term performance. Specifically, we found that the number of student correct and incorrect responses were not reliable predictors of long-term performance. This result is in contrast to most student-modeling efforts on predicting performance on the next response. Second, we argue that answering the question of whether a student will retain a skill is more useful for guiding decision making of intelligent tutoring systems (ITS) than predicting correctness of next response. We introduce an architecture that identifies two research topics that are meaningful for ITS decision making. Our experiments found one feature in particular that was relevant for student retention: the number of distinct days in which a student practiced a skill. This result provides additional evidence for the spaced practice effect, and suggests our models need to be aware of features known to impact retention.

## Keywords

Educational data mining, intelligent tutoring systems, student modeling, knowledge retention.

## 1. INTRODUCTION

The field of the educational data mining (EDM) has been focusing on predict correctness of the next student response for many years (e.g. .[2]). Very little work has been done with respect to longer-term prediction. Two common approaches for student modeling are knowledge tracing [5] and performance factors analysis [10]. Both of these approaches focus on examining past student performances, and predicting whether the student's next response will be correct or incorrect. The source of power for both of these techniques is the student's pattern of correct and incorrect responses. In fact, that input is the only piece of information knowledge tracing (KT) uses (beyond which skill the problem is associated with). KT observes whether the student responds correctly or not, and uses its performance parameters, guess and slip, to update its estimate of the student's knowledge. KT takes the form of a dynamic Bayesian network, where each time slice represents an item the student is working on.

Performance factors analysis (PFA) works similarly, and keeps track of the number of correct and incorrect responses the student has made on this skill. In addition, some versions of PFA also make use of an item difficulty parameter to account for item complexity. PFA takes the form of a logistic regression model, predicts whether the student will respond to an item correctly, and estimates coefficients for the number of correct and incorrect responses that maximize predictive accuracy.

A connection with student modeling is mastery learning. In a mastery learning framework, a student continues to practice a skill until it is "mastered." The exact definition of mastery varies, but typically involves recent student performance. For example, the KT framework suggests that the probability a student knows a skill exceeds 0.95, then the student has mastered the skill. The ASSISTments project (www.assistments.org) uses a simpler heuristic of three consecutive correct responses to indicate mastery.

However, there is evidence that strictly local measures of student correctness are not sufficient. Specifically, students do not always retain what they have learned. Aside from the psychology literature (e.g. [1,4,7,8]) there has been work within student modeling that demonstrated students were likely to forget some material after a delay. Qiu et al. [12] extended the Knowledge Tracing model, to take into account that students exhibit forgetting when a day elapses between problems in the tutor.

Researchers in the ITS field are currently using short-term retention as an indicator for mastery learning. However, for a cumulative subject like mathematics, we are more concerned with the ability of the students to remember the knowledge they learned for a long period of time. Pavlik and Anderson [9] studied alternative models of practice and forgetting, and confirmed the standard spacing effect in various conditions and showed that wide spacing of practice provides increasing benefit as practice accumulates, and less forgetting afterwards as well, which is consistent with classic cognitive science results [3].

## 2. PROBLEM AND APPROACH

Although the fields of student modeling and EDM have focused on short-term student performance, there is nothing inherent in student modeling or in EDM that requires such a focus. Conceptually, it is possible to construct models that predict other constructs of interest, such as whether the student will remember a skill after a period of time. Why would we want to construct such a model? We argue that whether a student will not only respond correctly on an item right away, the mastery approach used by KT, but whether the student will remember enough to respond correctly, after taking a break from working with the tutor, is a better definition of mastery. At best, it is unclear how to apply a short-term model such as KT or PFA for such a decision-making task. However, if we could build such a detector, we could

deploy it in a computer tutor and use it to decide when to stop presenting items to students. Perhaps a student who gets several items correct in a row, and masters the skill by traditional definition, will be predicted to not retain the skill and should receive additional practice.

The approach we use is straightforward: rather than attempting to predict every next student performance, instead we focus on student performances that occur after a long delay. In this way, even though we are not explicitly modeling the forgetting process, our student modeling approach captures aspects of performance that relate to student long-term retention of the material. It is reasonable that the field of student modeling did not start with long-term retention, as only a small minority of student practice opportunities take place after a long delay. Therefore, such restrictions would result in a too-small data set to train the student model parameters. However, with the advent of large educational sets, such restrictions become less relevant.

The data used in this analysis came from the ASSISTments system, a freely available web-based tutoring system for 4th through 10th grade mathematics (approximately 9 through 16 years of age). The system is mainly used in urban school districts of the Northeast United States. Students use it in lab classes that they attend periodically, or for doing homework at night.

We collected data from school year September 2010 to September 2011, which consisted of 15931 students who solved at least 20 problems within ASSISTments. We filtered out skills that have fewer than 50 students. As a result, we have 2,363,982 data records. Each data record is recorded right after a student answered a problem, and logged relevant information including the identity of the student, the problem identity and skills required to solve it, the correctness of the student's first response to this problem, the duration the student spent on this problem, and the timestamp when the student start and finish solving this problem.

For this task, we defined a student as retaining a skill if he was able to respond correctly after approximately a week. We instantiated a week as any duration between 5 and 10 days. and choose the time interval of 5-10 days as our objects to analyze. We randomly selected one fourth of the students as training data, which result in 27,468 final data records. Note that less than 5% of the data are relevant for training a model of student retention. Thus, this problem requires large data sets.

# 3. STUDENT RETENTION ANALYSIS

## 3.1 RQ1: Is student retention predictable?

To answer this question, we built a logistic regression model, using the 27,468 data points with delayed practice opportunities described previously. The dependent variable is whether the student responded correctly on this delayed outcome. We used user identity (user_id) and skill identity (skill_id) as factors (fixed effets) in this model. We used the following features as covariates, treating incorrect responses as a 0 and correct responses as a 1:

- *n_correct*: the number of prior student correct responses on this skill; This feature along with *n_incorrect*, the number of prior incorrect responses on this skill are both used in PFA models;
- *n_day_seen*: the number of distinct days on which students practiced this skill. This feature distinguishes the students who practiced more days with fewer opportunities each day from those who practiced fewer days but more intensely, and allow us to evaluate the difference between these two situations. This feature was designed to capture certain spaced practice effect in students data;
- *g_mean_performance*: the geometric mean of students' previous performances, using a decay of 0.7. For a given student and a given skill, use *opp* to represent the opportunity count the student has on this skill, we compute the geometric mean of students' previous performance using formula: *g_mean_performance(opp) = g_mean_performance(opp-1)*0.7 + correctness(opp)*0.3*. The geometric mean method allows us to examine current status with a decaying memory of history data. The number 0.7 is selected based on experimenting with different values.
- *g_mean_time*: the geometric mean of students' previous response time, using a decay of 0.7. Similar with *g_mean_performance*, for a given student and a given skill, the formula of the geometric mean of students' previous response time is: *g_mean_time(opp) = g_mean_time(opp-1)*0.7 + response_time(opp)*0.3*;
- *slope_3*: the slope of students' most recent three performances. The slope information helps capture the influence of recent trends of student performance;
- *delay_since_last*: the number of days since the student last saw the skill. This feature was designed to account for a gradual forgetting of information by the student;
- *problem_difficulty*: the difficulty of the problem. The problem_difficulty term is actually the problem easiness in our model, since it is represented using the percent correct for this problem across all students. The higher this value is, the more likely the problem can be answered correctly.

It is important to note that the features were computed across all of the data, not just the items on which the student had not practiced the skill for 5 to 10 days. For example, the n_correct feature is computed across *all* of the student practices on the skill, not just those practices with a 5 to 10 day delay interval. However, we only create a row in our data set for such delayed retention items (thus there are 27,468 rows). After training the model on the ASSISTments data, we got a $R^2$ of 0.25. Since this model fit represents training-data fit, it is optimistic. But the model fit is at least strong enough to conclude that student retention appears to be predictable.

The Beta coefficient values and p-values for each covariate are shown in Table 1.

In this table, the positive B values mean the larger the covariate is, the more likely the student respond to this problem correctly. To our surprise, the influence of the *n_correct* and the *n_incorrect* features are not reliably different than 0. The features *n_day_seen* and *g_mean_performance,* on the contrary, are reliable predictors of student retention. In other words, for predicting long-term retention, the number of days on which the student practiced the skill is important, as is his recent performance. This result is consistent with cognitive "spaced practice effect" result [11]. The raw number of correct and incorrect responses is not a meaningful

predictor. We expected that response time would be relevant to retention, due to its connection to automaticity and mastery [1].

| Covariate | B | *p-value* |
|---|---|---|
| ***n_correct*** | -0.003 | .330 |
| ***n_incorrect*** | -0.005 | .245 |
| ***n_day_seen*** | 0.055 | .000 |
| ***g_mean_performance*** | 0.813 | .000 |
| ***g_mean_time*** | 0.073 | .043 |
| ***slope_3*** | -0.033 | .444 |
| ***delay_since_last*** | -0.015 | .182 |
| ***problem_difficulty*** | 5.926 | .000 |

Table 1. Parameter table of covariates in Model1.

From the likelihood ratio tests of the training set, we found that the *skill_id* and *user_id* are also both important features in this model. This indicates that student performance on retention items varies by skill and by student. It is tempting to claim that retention varies by student, but this claim is premature as the user_id factor models student performance on retention items. However, such performance is composed of how well the student learned the material as well as how much of that knowledge was retained. A student could have a strong memory, but if he never learned the material his user_id factor would be low. Therefore, user_id does not solely represent retention.

To strengthen the results, we built test set to validate the model. Since the users of testing set are different from those of the training set, we cannot look up user parameters directly for users in the testing set. Instead, we use the mean value of user parameters of the model as an approximation of the user parameter in the testing set. We also did the same thing for the skills that only appear in the testing set.

The R2 of this model on the testing set is 0.17, indicating a reasonable model fit in-line with other attempts at using PFA

## 3.2 RQ2: Does forgetting vary by student?

We would like to separate the impact of the user_id feature into student knowledge and student retention. To accomplish this task, first, we started from the logistic regression model that we used in section 3.1, removed the factor *user_id* and substituted a covariate *non_1st_pcorrect*. The feature *non_1st_pcorrect* is the percent of a student's non-first attempts of the day that are correct. The intuition is that a student's first attempt on a skill each day is the one that is most affected by retention. By considering the student's overall performance, but excluding these items, we are estimating the student's proficiency in the domain in a way that is less contaminated with forgetting, and is thus a purer estimate of the student's knowledge. We trained this model on the same data as the previous model. The feature *non_1st_pcorrect* has an estimated Beta coefficient of 3.878, with a p-value 0.000. We got an $R^2$ of 0.210 on the data, which is a reasonable model fit. The difference in model fit is caused by the substituting the percent correct, on non-first encounters, for user_id.

We were curious as to the cause of this difference in model fits, and investigated the residuals from our model. The question is whether the residual was systematic, and could be predicted by user_id. We fit a general linear model with *user_id* as a random effect, and the residual as the dependent variable. The $R^2$ of this model is 0.235. Thus, the residual in our model, after accounting for student overall percent correct in contexts where forgetting was minimal, does vary systematically by user_id. Thus it appears that there is some construct beyond performance, such as forgetting, that varies by student.

Although it is tempting to claim this term represents student forgetting, it is necessary to validate the construct [6] we have modeled. To test whether we have modeled retention, we first extracted the student random effects from our GLM. We then computed the correlation between that term, and each student's difference in performance between the first and second question on a skill that occurs each day. Our belief is that this difference in performance is related to student forgetting, since a large increase in performance from the first to the second item suggests the student is remembering old information. Unfortunately, the correlation between these terms was negligible, so we are still searching for what our per-student effect is actually modeling.

# 4. CONTRIBUTIONS

This paper makes three main contributions. First, the mastery learning notion is expanded to take into account the long-term effect of learning. In comparison to the traditional view that Corbett and Anderson brought up in their seminal work [5], which looks at only the immediate knowledge, this paper looks at broader notion of knowing a skill.

The second contribution this paper makes is extending the PFA model [10] with features that are likely to be relevant for retention. Most prior work has focused on concepts such as item difficulty or amount of assistance required to solve a problem. However, those features focus on student performance and properties of items, not on broad characterizations of performance. Our study confirmed that the long-term knowledge appears to vary by skill, and possibly by student. In addition, the number of days on which a student practiced a skill is relevant, and could be an important feature in directing ITS decision making to enhance retention. This result confirms the spaced practice effects in a larger scope; also we found that the number of correct responses seem to be not so important in predicting knowledge retention.

The third contribution this paper makes is on discovering a new problem that is actionable by ITS. Previous student models focus on estimating student current knowledge, which is powerful for EDM, and an efficient use of data for testing a model, but provides limited guidance for tutorial decision making. This paper proposed a diagram of ITS action cycle that can be used to discover new problems in the EDM field that can lead to higher mastery learning rate in ITS systems.

One goal of EDM is to address questions that are relevant for tutorial decision making of ITS. Currently, many ITS simply present a sequence of problems and evaluate student performance right after the student finished these problems to see if the student mastered the given skill. This process does not have the mechanism for the system to review students' knowledge after a time period, nor know about students' long term performance. It is dangerous for ITS to promote a student on the basis of short term performance. We propose the follows diagram shown in

Figure 1, which allows ITS to aim for students long-term mastery learning.

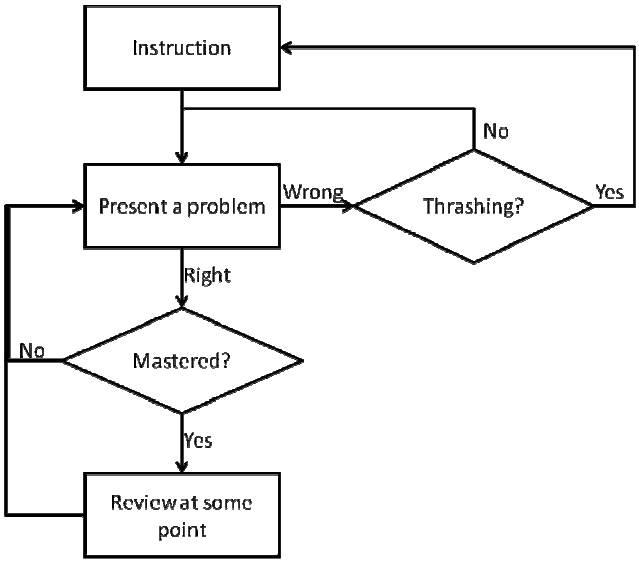



Figure 1. Enhanced ITS mastery learning cycle

This paper focuses on the diamond on the left side, whether the student has mastered a skill. Rather than using local criteria to decide whether mastery has occurred, we trained a model to decide mastery based on predicted performance. Beyond this EDM work, some review mechanism for ITS seems warranted, as for cumulative domains, such as mathematics, ensuring that student's have retained their knowledge is critical. Correspondingly, we have added a review mechanism to ASSISTments [13].

Another interesting EDM problem is the diamond on the right: when is a student likely to fail to master a skill in a timely manner and (statistically) exhibit negative behaviors such as gaming? We have made progress on this problem as well, and dubbed the phenomenon "thrashing." If a student is unlikely to master a skill via problem solving, it is essential to do something else, such as peer tutoring, having the teacher intervene, or providing instruction within the ITS.

What we like about both of these problems is that they are rich challenge problems for EDM, and provide actionable information for ITS to make use of in their decision making. If a student appears likely to retain a skill, it is probably not necessary to keep presenting items. If a student is likely to not master a skill, it is probably not productive to keep presenting problems.

## 5. FUTURE WORK AND CONCLUSIONS

There are three questions that we are interested in exploring. First, do students vary in how quickly they forget? Our first attempt at teasing apart the user_id factor gave inconclusive results, but this area is important enough to warrant further study. Another issue that we are interested in addressing is what are additional features that relate to forgetting? The field of psychology is rich in ideas, but there has been little existing work in student modeling.

Finally, we would like to deploy this model to a working ITS in the field. On one hand, this can help verify the model; on the other hand, this could be used to improve the ITS systems to help student achieving long-term mastery learning.

This paper present an ITS mastery learning diagram, which brings up useful problems in EDM that needs more work. In this paper we concentrate on estimating student knowledge retention and discovered some useful features for this task. Also, we were able to conclude student long-term performance is predictable, even when a student's ability to remember a skill comes in to play.

## 6. REFERENCES

[1] Anderson, J.R., Rules of the Mind. Lawrence Erlbaum (1993).

[2] Beck, J.E., et al. Predicting student help-request behavior in an intelligent tutor for reading. Ninth International Conference on User Modeling, (2003), Johnstown, PA.

[3] Cain, L.F. and Willey, R.D.V, The effect of spaced learning on the curve of retention. Journal of Experimental Psychology, Vol 25(2), (Aug 1939), 209-214. doi: 10.1037/h0054640.

[4] Cepeda, N.J., et al. Distributed practice in verbal recall tasks: A review and quantitative synthesis. Psychological Bulletin, Vol 132(3), (May 2006), 354-380. doi: 10.1037/0033-2909.132.3.354.

[5] Corbett, A.T. and J.R. Anderson, Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, (1995), 4: p. 253-278.

[6] Crocker, L. and J. Algina, Introduction to Classical and Modern Test Theory. (1986), Fort Worth: Harcourt Brace Jovanovich College Publishers. 482.

[7] Ebbinghaus, H., Memory: A Contribution to Experimental Psychology. (1885), New York: Teachers College, Columbia University.

[8] George B.S. and John A.E., Knowledge Taught in School: What Is Remembered?, Review of Educational Research Summer, (1994) vol. 64 no. 2 253-286

[9] Pavlik, P.I. and J.R. Anderson, Practice and Forgetting Effects on Vocabulary Memory: An Activation-Based Model of the Spacing Effect. Cognitive Science, (2005). 29(4): p. 559-586.

[10] Pavlik, P.I., H. Cen, and K.R. Koedinger, Performance factors analysis—A new alternative to knowledge tracing., in Proceedings of the 14th International Conference on Artificial Intelligence in Education, V. Dimitrova and R. Mizoguchi, Editors. (2009): Brighton, England.

[11] Perruchet, P. (1989), The effect of spaced practice on explicit and implicit memory. British Journal of Psychology, 80: 113–130. doi: 10.1111/j.2044-8295.1989.tb02306.x

[12] Qiu, Y., et al. Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing. in Fourth International Conference on Educational Data Mining. 2011.

[13] Wang, Y. and N.T. Heffernan. Towards Modeling Forgetting and Relearning in ITS: Preliminary Analysis of ARRS Data. in Fourth International Conference on Educational Data Mining. 2011.

# POSTERS

# A promising classification method for predicting distance students' performance.

Diego García-Saiz
Universidad de Cantabria
Avda. Los Castros s/n
Santander, Spain
diego.garcia@unican.es

Marta Zorrilla
Universidad de Cantabria
Avda. Los Castros s/n
Santander, Spain
marta.zorrilla@unican.es

## ABSTRACT

Predicting the students' performance is still a challenging task despite being one of the oldest and most popular applications of data mining in education. One of the problems encountered when analyzing data from e-learning platforms is that it presents statistical outliers as a consequence of how students work in online courses. It causes that classifiers are built with less accuracy than desired. To solve this problem we propose a new method to eliminate outliers as previous step to build the classifier. In this work we describe our meta-algorithm and compare its performance with respect to several well-known classification techniques. The comparison is evaluated in terms of accuracy, true positive and true negative rate. The results obtained shows that our approach produces more accurate models.

## 1. INTRODUCTION

Since the advent of learning platforms, their use in educational centers has been constantly growing. Unlike traditional teaching, one of the advantages which these systems have is they store a huge quantity of data which, adequately analysed, can help both instructors and students: instructors can discover information to evaluate the teaching-learning process [6]; and, students can receive suitable feedback about their dedication in the course [4] and recommendations in order to achieve the learning objectives [5].

In the literature, there are several works which compare classification techniques using educational datasets but none of them analyse the data distribution in order to detect outliers [3; 1]. In our experimentation, we have detected that despite the data is clean (free of human errors), there are instances which can be considered as outliers in the statistical sense (e.g. students with one learning session can pass the course and students with a high time spent in the course fail) and these must be eliminated in order to improve the classifier accuracy. Therefore, we have designed and implemented a meta-algorithm which carries out both tasks: pre-processing and modelling. Our goal is to offer this meta-algorithm to educational community so they can build more accurate classification models.

Next, we describe our meta-algorithm and compare its accuracy, TPR and FPR with respect to that obtained with 5 of the most frequently used classification algorithms [7].

## 2. META-ALGORITHM DESCRIPTION AND EXPERIMENTATION

Before explaining our meta-algorithm, we show why a removal outlier phase is necessary. In this experimentation we work with a course hosted in Blackboard and taught in 2009, 2010 and 2011 at University of Cantabria, entitled "Introduction to Multimedia". We generated two different datasets. Dataset1 includes the following attributes: total time spent, number of sessions, average time per session, average time per week and average number of sessions per week. Dataset2 includes the same attributes but aggregated by month and by tool (content-page, forum and mail). For instance, total time spent in January reading content-pages. Both data sets have 194 instances (one per each student) and they don't contain missing data. We show in Figure 1, the distribution of students who passed and failed with respect to the total time spent and the total number of sessions carried out. As can be observed, there are a few students who failed despite spending a lot of time in the course and often connect, and students who passed (see red squares between lines draw in Figure 1) with an average time and a number of sessions similar to those who failed. The reasons of this bad behavior is intrinsic to the way of working in the web. Students connect to the e-learning platform and, after some clicks, they open another URL out of the course, and work in parallel in both. So that the total time does not correspond with the total time of work. And viceversa, students who work hardly but in a disconnected mode since they download the materials. Thus, if we want to improve our classification models we have to minimize the effect of this problem.

Our meta-algorithm works as follows. First, it carries out a correlation study and removes those attributes which are dispensable. Next, it builds a two-class classifier with all instances and determines which are the incorrectly classified instances of both classes. Next, it calculates the prototype of each class and the euclidean distance of each instance to its prototype. Then it obtains the average distance in each class, $ED$, and chooses that class which has a bigger value of $ED$. Once chosen the class with higher $ED$, named $K$, the incorrectly classified instances of the $K$ class are selected and the meta-algorithm carries out a clustering using Kmeans in order to separate these instances in two groups. Built the two clusters, it calculates the centroid in both clusters and names $N$ and $M$ respectively. Then it removes from the training sets the instances belonging to the cluster whose centroid has a larger euclidean distance to the

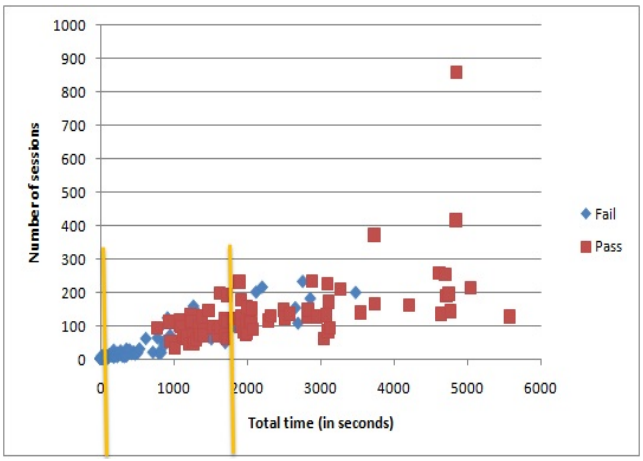



Figure 1: Class distribution according to total time and number of sessions

Table 1: Comparison of models obtained with Dataset1

| Alg. | Orig. | | | Malg | | |
|---|---|---|---|---|---|---|
| | Acc. | TPr | TNr | Acc. | TPr | TNr |
| J48 | 81.44 | 95.62 | 71.68 | 82.47+ | 96.30+ | 72.57+ |
| NB | 77.84 | 64.20 | 87.61 | 83.51+ | 83.95+ | 83.18- |
| OneR | 78.87 | 87.65 | 72.57 | 78.35- | 86.42- | 72.57- |
| RTree | 78.35 | 75.31 | 80.53 | 80.93+ | 91.36+ | 73.45- |
| JRip | 81.95 | 92.59 | 74.34 | 85.05+ | 97.53+ | 76.11+ |
| Avg. | 79.69 | 83.07 | 77.34 | 82.06+ | 91.11+ | 75.57- |

instances of the selected $K$ class. That means, it eliminates the bad-classified instances, those that have a more irregular distribution with respect to the distribution of the instances set of $K$ class. After removing the instances considered as outliers from the training sets using 10-CV, it builds the final classifier. Of course, outliers are only removed from the training sets, leaving the test sets with all initial instances. Table 1 shows the results of applying our meta-algorithm (Malg) using 5 different classification algorithms on Dataset1. As can be observed, our meta-algorithm improves in accuracy (Acc.) and TPrate (TPr) the results obtained with respect the original (Orig.) classification algorithm without removing instances, except with OneR. TNrate (TNr) is sometimes lower due to the fact that the negative class was chosen by the meta-algorithm in the preprocessing phase in this dataset. In particular, the instances with negative class eliminated correspond to the blue diamonds behind the second vertical line in Figure 1. We obtain similar results when we use Dataset2 (see Table 2). Using this dataset, our meta-algorithm improves the results of accuracy in all cases and sometimes worsens the TNrate with respect to the original classification algorithm.

## 3. CONCLUSIONS AND FUTURE WORK

The experimentation carried out in this work allow us to conclude that the preprocessing tasks generally improve the classification models when data sets suffer from statistical outliers. In our case study this is traduced to find students with both, a high total time spent and a high number of sessions in a virtual course who, at the end, failed.

Table 2: Comparison of models obtained with Dataset2

| Alg. | Orig. | | | Malg | | |
|---|---|---|---|---|---|---|
| | Acc. | TPr | TNr | Acc. | TPr | TNr |
| J48 | 87.11 | 96.29 | 80.53 | 87.63+ | 95.06- | 82.30+ |
| NB | 77.84 | 65.43 | 86.73 | 80.93+ | 87.65+ | 76.11- |
| OneR | 86.08 | 97.53 | 77.87 | 88.66+ | 100.00+ | 80.53+ |
| RTree | 82.47 | 76.54 | 86.73 | 87.63+ | 87.65+ | 87.61+ |
| JRip | 86.08 | 93.83 | 80.53 | 88.14+ | 98.77+ | 80.53= |
| Avg. | 83.92 | 85.92 | 82.48 | 86.59+ | 93.87+ | 80.53- |

The meta-algorithm implemented allows us to build more accurate classifiers, so instructors can predict better the student's performance of their courses and improve the teaching process. Nevertheless we are working in some improvements such as extending the meta-algorithm in order to work with a multi-valued class (more than two) or adapting it for working with multi-instance predictors, which will allow us to combine instances of courses whose organization is different and to obtain a common model for all of them. Finally, we implement a new template for our ElWM tool [2] which uses this meta-algorithm.

Out of educational data mining context, this meta-algorithm offers an opportunity to improve any classification model which presents statistical outliers in its training datasets.

## 4. REFERENCES

[1] M. Cocea and S. Weibelzahl. Cross-system validation of engagement prediction from log files. In *EC-TEL*, pages 14–25, 2007.

[2] D. García-Saiz and M. E. Zorrilla. E-learning web miner: A data mining application to help instructors involved in virtual courses. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *EDM*, pages 323–324. www.educationaldatamining.org, 2011.

[3] W. Hämäläinen and M. Vinni. Comparison of machine learning methods for intelligent tutoring systems. In M. Ikeda, K. Ashley, and T.-W. Chan, editors, *Intelligent Tutoring Systems*, volume 4053 of *Lecture Notes in Computer Science*, pages 525–534. Springer, 2006.

[4] A. A. Juan, T. Daradoumis, J. Faulin, and F. Xhafa. A data analysis model based on control charts to monitor online learning processes. *Int. J. Bus. Intell. Data Min.*, 4:159–174, July 2009.

[5] X. Li, Q. Luo, and J. Yuan. Personalized recommendation service system in e-learning using web intelligence. In *Proc. of the 7th international conference on Computational Science, Part III*, pages 531–538, 2007.

[6] C. Romero and S. Ventura. Educational data mining: A review of the state of the art. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(6):601–618, 2010.

[7] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14:1–37, December 2007.

# Analyzing paths in a student database

Renza Campagni, Donatella Merlini, Renzo Sprugnoli
Dipartimento di Sistemi e Informatica
viale Morgagni 65, 50134, Firenze, Italia
[renza.campagni,donatella.merlini,renzo.sprugnoli]@unifi.it

## 1. INTRODUCTION

In the recent literature, several data mining models have been proposed to understand and improve the educational performance and assessment of a student learning process. For example, in [1] the authors illustrate a classification model to investigate the profile of students which most likely leave university without ending their study; in [2], the author uses association rule mining for assessing student results. Data mining techniques have been also applied in computer-based educational systems (see, e.g., [3]). In our research we present a data mining approach to analyze a database containing information about students, in particular, their personal but anonymous data and their exams. We consider the *path* of a student, that is, the way the student implement her or his exams over the degree-learning time: a student can take an exam immediately after a course, the *ideal* choice, or later.[1] The aim of this work is to understand how this order affects the performance of the students in terms of graduation time and final grade. See [4] for a recent research in the field of choices in learning. We consider the *ideal path*, i.e., the path of a student which has taken each examination just after the end of the corresponding course, without delay. Therefore, the ideal path matches the curriculum settled by the academic degree. The path of a generic student is then compared with the ideal path by using two different approaches: the first one uses the *Bubblesort* distance while the second is based on the computation of the *area* between the two paths. Students who have taken the exams in the same order, that is, students with the same path, can have different final grade and graduation time. The idea is to understand if there exists a relation between these distances and the success of students. If the students having small distances achieve good performance, then we may conclude that the academic degree is well structured but if there exist many good students with large distances, then can mean that the organization should be modified. Once the distances have been computed, they can be inserted in the database, as new attributes of each student, and a clustering analysis can be performed, for example by using the `K-means` implementation of `WEKA` (see, e.g., [5]). By using this methodology on our database, with both approaches and `K` $= 2$, we obtained two clusters characterized by *small* and *large* distances. The first one corresponds to the group of students who graduated relatively quickly and with high grades; the second cluster corresponds to students who obtained worst results. Our analysis shows that the more students follow the order given by the ideal path the more they get good performance in terms of graduation time and final grade. In conclusion, no student with a large distance achieves good results; we can conclude that the academic degree under consideration was well scheduled.

[1]We refer to an organization which allows students to take an exam in different sessions after the end of the course, as in Italy. A drawback is that students end up graduating with a significant delay. Some constrains between exams can be fixed in order to force students to take exams in a specific order, however, usually students have many degrees of freedom.

## 2. THE METHODOLOGY

We consider a database containing the data of $N$ students, each student characterized by a sequence of $n$ exams identifiers. We consider a particular path $\mathcal{I} = (e_1, e_2, \cdots, e_n)$, the *ideal path*, corresponding to a student which has taken every examination just after the end of the corresponding course, without delay. Without loss of generality, we can assume that $e_i = i$, $i = 1, \cdots, n$, that is, $\mathcal{I} = (1, 2, \cdots, n)$. The path of a generic student $J$ can be seen as a sequence $\mathcal{J} = (e_{J_1}, e_{J_2}, \cdots, e_{J_n})$ of $n$ exams, where $e_{J_i}$, $i = 1, \cdots, n$, is the identifier of the exam taken by the student at time $i$. Therefore, $\mathcal{J}$ can be seen as a permutation of the integers 1 through $n$. In order to understand how the order of the exams affects the final result of students, we compare a path $\mathcal{J}$ with $\mathcal{I}$ by using two different approaches. The first approach uses the *Bubblesort distance*, which is defined as the number of exchanges performed by the Bubble sort algorithm to sort an array containing the numbers from 1 to $n$. The number of exchanges, bounded above by $n(n-1)/2$, can be computed easily since it is exactly the number of inversions in the permutation. Our second approach concerns the graphical representation of the paths; we represent them in the integer lattice, the $x$-axis denotes the number of exam and the $y$-axis the exam identifier, according to the order of the ideal student. The ideal path is defined by the sequence of points $\tau_{\mathcal{I}} = ((0, e_0), (1, e_1), (2, e_2), \cdots, (n, e_n), (n+1, e_{n+1}))$, where $e_0 = 0$ denotes the starting point of the path and $e_{n+1} = n + 1$ denotes the final examination taken last by all students. Therefore, $\tau_{\mathcal{I}}$ can be represented as the bisecting line of the first quadrant. The path of a generic student $J$, is then represented by a broken line corresponding to the sequence of points $t_J = ((0, e_{J_0}), (1, e_{J_1}), (2, e_{J_2}), \cdots, (n, e_{J_n}), (n + 1, e_{J_{n+1}}))$. By convention, we have $e_{J_0} = 0$ and $e_{J_{n+1}} = n+1$ for every student $J$, that is, the resulting trajectory begins at $(0, 0)$ and finishes in the point $(n + 1, n + 1)$ . We then compare a path $\mathcal{J}$ with $\mathcal{I}$ by computing the area $\mathcal{A}_{\mathcal{J},\mathcal{I}}$ between

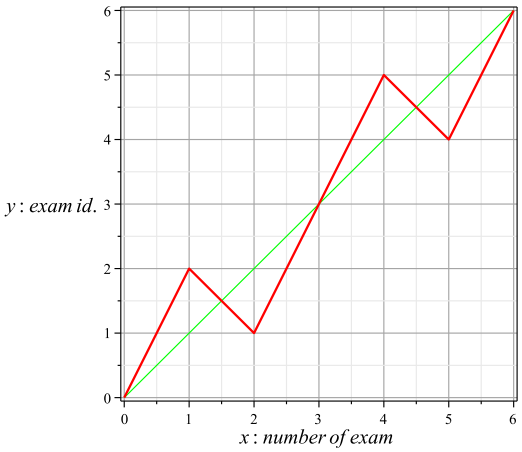



Figure 1: The path $(2, 1, 3, 5, 4) : \sigma(\mathcal{J}) = 2$ and $\mathcal{A}_{\mathcal{J},\mathcal{I}} = 3$.

$\tau_{\mathcal{I}}$ and $t_J$. In Figure 1 we illustrate the path $(2, 1, 3, 5, 4)$ and its distances from the ideal path $(1, 2, 3, 4, 5)$.

## 2.1 Cluster analysis

The database we analyze contains data of students in Computer Science at the University of Florence beginning their studies during the years 2001-2004 and graduated up to now. The academic degree in Computer Science at the University of Florence is structured in three years (Laurea triennale) and students can choose among different curricula. Every year is organized in two semesters; there are several courses in each semester and at the end of a semester students can take their examinations. Exams can be taken in different sessions during the year and students can try to pass their exams in any of these sessions, after the end of the course. In the years under consideration, no constrains between exams were fixed, so students could take their exams almost in any order. For each student, the database contains the identifier of the student, the grade obtained at the high school level, the year of enrollment at the university, the date and the mark of final examination and the sequence of exams; each exam is described by an identifier, a date and a grade. We considered students belonging to two slightly different curricula: databases and distributed systems. In particular, we analyzed the paths of $N = 100$ students characterized by a sequence of $n = 25$ exams. For each curriculum we computed the ideal path through an important pre-processing phase, which allowed us to identify the semester in which courses were originally hold by the teacher. In fact, the original database did not contain the information about the semester, which is fundamental for our purposes. In the ideal path, courses relative to the same semester were sorted by taking into account the preference of students. Therefore, we obtained two different ideal paths of length 25. Then, for each student $J$ of both curricula we computed the distances $\sigma(\mathcal{J})$ and $\mathcal{A}_{\mathcal{J},\mathcal{I}}$ from the ideal path and inserted these values into two fields `Bubblesort` and `Area` of the database. We tried to sort our data according to both fields and we found some pairs of paths having values of `Bubblesort` and `Area` in reverse order. Therefore, these two distances are not completely equivalent; however, this difference seems not to be important for the clustering analysis, as we will see later. To understand how the order of the exams affects the path of the students, we have performed several tests by using the `K-means` implementation of `WEKA`. We first analyzed the paths of students separately for the two curricula. In particular, in both cases, we obtained significant result with $K = 2$ and by selecting as clustering attributes the graduation time, `Time`, expressed in days, and the final grade, `Grade`, an integer between 66 and 110. In fact, with these parameters we can see that students are well divided into two groups: the group of students who graduated relatively quickly and with high grades and the group of students who obtained worst results, respectively. Luckily, we observed that students in the first group are characterized by *small* values of `Bubblesort` and `Area` while students in the second group have *larger* values. Our analysis shows also that the path of a student seems not to be affected by the results achieved at the high school level. We performed similar tests by adding the distance values as attributes of clustering, and we obtained two more distinct clusters, which divide students more precisely in terms of `Time`, `Grade` and `Bubblesort` (or `Area`) distance. We finally analyzed together the students belonging to the two curricula obtaining 2 clusters with the following characteristics:

| Attribute | Full data | Cluster 1 | Cluster 2 |
|---|---|---|---|
| Bubblesort | 89.7 | 71.8 | 119.2 |
| Area | 121.5 | 96.8 | 162.2 |
| Time | 2156.7 | 1841.9 | 2678.1 |
| Grade | 98 | 101 | 94 |

This result confirms that, regardless of the curriculum, the more students follow the order given by the ideal path, the more they obtain good performance in terms of graduation time and final grade. We point out that we obtained similar results by using either distance. In conclusion, the methodology proposed in this research can be used to valuate the organization of an academic degree in terms of the scheduling of the courses.

## ACKNOWLEDGEMENTS

The authors would like to thank Dino Pedreschi for interesting discussions about the model presented in this paper.

## 3. REFERENCES

[1] K. Daimi and R. Miller. Analyzing student retention with data mining. In *Proceedings of the 2009 International Conference on Data Mining*, pages 55–60, 2009.

[2] R. Damaševičius. Analysis of academic results for informatics course improvement using association rule mining. In *Information Systems Development*, pages 357–363. Springer, 2010.

[3] C. Romero, J. R. Romero, J. M. Luna, and S. Ventura. Mining rare association rules from e-learning data. In *The 3rd International Conference on Educational Data Mining*, pages 171–180, 2010.

[4] H. Soundranayagam and K. Yacef. Can order of access to learning resources predict success? In *The 3rd International Conference on Educational Data Mining*, pages 323–324, 2010.

[5] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

# Analyzing the behavior of a teacher network in a Web 2.0 environment

Eliana Scheihing
Universidad Austral de Chile
Casilla 567, Valdivia, Chile
eliana.scheihing@gmail.com

Carolina Aros
Universidad Austral de Chile
Casilla 567, Valdivia, Chile
carolinaaros@gmail.com

Daniel Guerra
Universidad Austral de Chile
Casilla 567, Valdivia, Chile
mallium@gmail.com

## ABSTRACT
In this paper, we analyze the interactions in a Web 2.0 platform of teachers from 60 schools integrating the Kelluwen network. This network is a part of a project aimed to introduce the Web 2.0 as a pedagogical tool for the development of socio-communicative skills in middle school students from Southern Chile. Clustering analysis was performed to understand the behavior of teachers in the network. The results allow us to distinguish two major groups. One characterized with a longer execution time of pedagogical activities -compared with the mean case- and the other one characterized by rating participation of their colleagues more frequently.

## Keywords
Clustering analysis, teacher network, Web 2.0

## 1. INTRODUCTION
The increasing number of on-line services supporting teacher activities requires the analysis of their impact on targeted users. There is little information on the use that teachers make in these environments [1]. In the particular case of a teacher network, one of the challenges is to discover the factors that allow the network to grow over the time. Kelluwen network is part of a project aimed to introduce the Web 2.0 as a pedagogical tool for the development of socio-communicative skills, in middle school students from Southern Chile [3]. The network is supported by an on-line platform that provides several services related to the use, development, evaluation, execution and sharing of collaborative didactical designs. In each of these activities, the teachers can interact on the platform with their students, other colleagues and other classes (*twin* classes) which are executing the same didactical design. For instance, in the managing module teachers register the beginning and the end of each activity performed with their students, in the worklog teachers communicate with their students. In the Kelluwen wall teachers interact with other colleagues talking about the shared didactical experiences, etc. The last version of the platform is available at http://www.kelluwen.cl/app. In this work, we perform a clustering analysis to discover patterns in teachers according to the interaction activities that they undergo in the Kelluwen platform. Clustering is an unsupervised learning model usually applied in educational data mining for discovering groups of users in the context of e-learning environments [1]. This study applies Latent Class Analysis (LCA) [2], and also the classical K-means clustering algorithm, using procedures of R statistical in both cases. This is a first study of the interactions in this teacher network which intends to discover factors that would allow the Kelluwen community of practice to grow over the time.

## 2. DATA ANALYSIS AND RESULTS
The data from Kelluwen network of teachers executing a didactical design during year 2011 were used in this study. From this, those who realised less than 20% of any didactical design were excluded. Finally, 55 teachers were considered of a total of 68 teachers registered in the network. We identified nine measures which are related to the principal interaction activities of the teachers in the platform. A first measure is the number of didactical designs executed for each teacher, named NºDD in the analyses. From the *Management module* we computed the mean and standard deviation of time between the beginning of two sequential activities over the didactical designs executed by a teacher (Mean time and SD time respectively). At the *Worklog module*, the teacher interacts with their students and with colleagues from twin classes. Three related measures are defined in this case: Number of messages, number of answers to messages, and rating of messages that a teacher has written in the Worklog, during all the didactical designs executed by her (named Worklog messages, responses and ratings respectively). At the *Kelluwen wall* the teacher interact with their colleagues sharing her experiences in the execution of didactical designs. Three related measures are defined in this case: Number of messages, number of answers to messages, and rating of messages that a teacher has written in the Kelluwen wall, during all the didactical designs executed by her. We named these variables Wall messages, responses and ratings respectively in the analyses.

### 2.1 K-means analysis
To calculate the appropriate number of cluster, we plot the within-groups sum-of-squares by number of clusters extracted. The plot in Figure 1 suggests K=5 or 6.

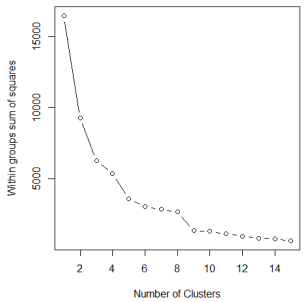



**Figure 1: sum of squares within groups as a function of number of clusters in K-means clustering**

Using the kmeans function from R, we compute the clustering with both values of K and we plot the obtained clusters jointly with the projection of original variables (red rows) in the principal component plane. Figure 2 shows the obtained clusters for K=5. There is one main group (4) with 43 individuals without distinctive features. Interestingly, two groups follow a more defined pattern: one group (3) of five individuals who spend more

time between sequential activities, and another group (2) of 3 individuals that use the social services of the platform to interact with other teachers. Two singleton groups are 1 and 5 which correspond to two teacher characterized by an intensive use of the *Virtual worklog.*

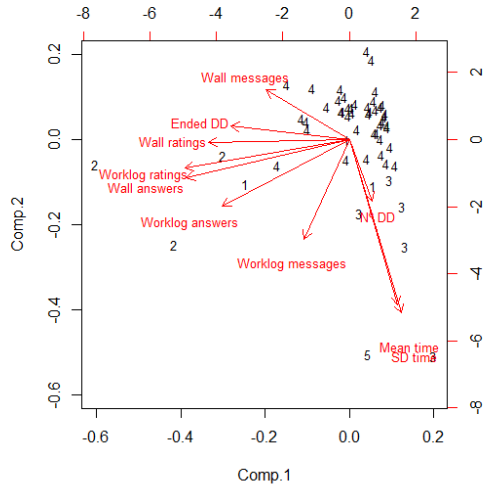



**Figure 2: K-means cluster analysis with k=5**

## 2.2 Latent cluster analysis

Following [4], we use the function mclustBIC from mcluster package of R to compute different models for latent cluster analysis. Two optimal models satisfying a maximum Bayesian Information Criterion (BIC) are obtained (see Figure 3). Both are multivariate models with independent variables of different variances and equal shape (VEI), with 3 or 4 components in the mixture. The values of BIC are -1943.008 and -1943.832. We select the model with 4 component because is more comparable with the k-means analysis.

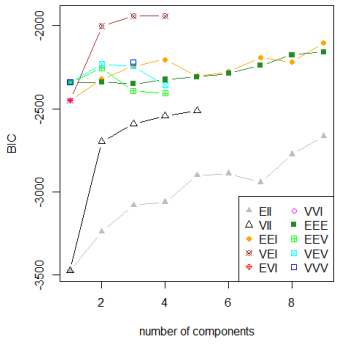



**Figure 3: BIC as a function of components in LCA**

The corresponding clusters are computed with the mclustModel function, which gives a probability to each individual to be extracted from each cluster. We consider values upper 0.5 to assign a cluster to each individual. The resulting clusters are plotted jointly with the projection of original variables (red rows), in the principal component plane in Figure 4. We observe two main groups (2 and 4), with 34 and 12 individuals respectively, without distinctive features, and two other groups of teachers that follow a more defined pattern: one group (3) of three individuals who spend more time between sequential activities, another group (1) of 6 individuals that use the social services of the platform to interact with other teachers.

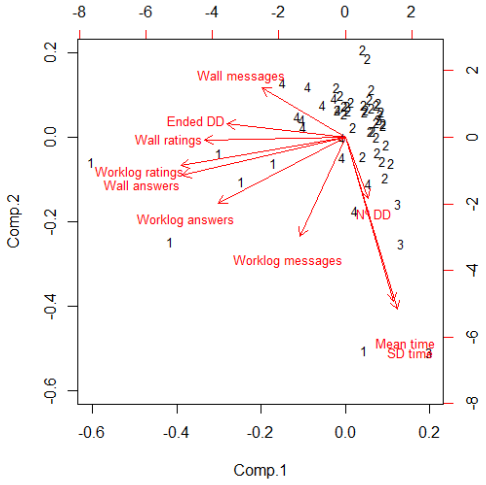



**Figure 4: Latent cluster analysis with 4 components.**

## 3. CONCLUSION AND DISCUSSION

The Kelluwen network is supported by a Social Web collaborative learning platform which contains meaningful information enclosed in their system dynamics. In this work we performed initial cluster analyses using the teacher interactions in the platform to characterize their behavior. The results in both analyses show a major group of teachers more or less homogeneous and two secondary groups with some key features. One of them characterized with a longer execution time of didactical activities -compared with the mean case-, and the other one characterized by rating participation of their colleagues more frequently. Furthermore in the principal component plane, both features are represented as orthogonal. This finding allow us to hypothesize that the more the teacher uses the Web 2.0 services of the Kelluwen platform, the more regular is her behavior executing the didactical design.

## ACKNOWLEDGMENTS

This work was partially funded under the code FONDEF D08i-1074 from CONICYT.

## REFERENCES

[1] B. Xu and M. Recker, «Understanding Teacher Users of a Digital Library Service: A Clustering Approach», *Journal of Educational Data Mining*, vol. 3, n°. 3, 2011.

[2] J. Magidson and J. Vermunt, «Latent class models», in *The Sage Handbook of Quantitative Methodology for the Social Sciences*, Sage Publications, 2004, pp. 175-198.

[3] E. Scheihing, L. Cárcamo, M. Arancibia, P. Contreras, and D. Guerra, «Kelluwen, estrategias para desarrollar competencias socio comunicativas usando servicios de la web 2.0», *X Encuentro Internacional Virtual Educa 2009.*

[4] D. Haughton, P. Legrand, and S. Woolford, «Review of Three Latent Class Cluster Analysis Packages: Latent Gold, poLCA, and MCLUST», *The American Statistician*, vol. 63, n°. 1, pp. 81-91, feb. 2009.

# Automated Detection of Mentors and Players in an Educational Game

Fazel Keshtkar
University of Memphis
Institute for Intelligent Systems
Memphis, TN, USA
fkshtkar@memphis.edu

Brent Morgan
University of Memphis
Institute for Intelligent Systems
Memphis, TN, USA
bmorgan2@memphis.edu

Arthur Graesser
University of Memphis
Institute for Intelligent Systems
Memphis, TN, USA
graesser@memphis.edu

## ABSTRACT

Automatically identifying the various roles (e.g., mentor, player) in multi-party collaborative chat is a challenging task. To better understand the conversational demands of mentors and players, this paper investigates the dynamics and linguistic features of multi-party chat in the context of an online educational game. In this paper we introduce a novel computational linguistics method using a machine learning algorithm to automatically classify utterances of players and mentors in a serious game, where players act as interns in an urban planning firm and discuss their ideas about urban planning and environmental science in written natural language. Our results are promising and our model can be extended to any multi-party environment that leaders (Mentors) are needed to be distinguished based on their conversation.

## Keywords

serious game, natural language processing, machine learning, multi-party chat

## 1. INTRODUCTION

Individuals in a collaborative learning environment often adapt to specific roles, whether organically or directly assigned. Automatic role detection would provide a crucial step in understanding the impact these roles have in collaborative learning dynamics [2]. Multi-party chat presents an especially challenging task, as the tone is conversational, and distinguishing between roles is relatively difficult (as opposed to email, for example).

Previous research suggests that humans quickly infer a speaker's intentions, often within the first few words of an utterance [3]. Thus, we selected UNIGRAMS and BIGRAMS as our units of analysis. Our results also show that UNIGRAMS perform much better than BIGRAMS. In our automated approach, we investigated the problem as supervised learning in set of features and then we used machine learning algorithms to learn the parameters of the model from annotated training data.

In this paper, we analyzed four chat room conversations between 21 players and two mentors as they interacted within the epistemic game *Urban Science* [4].

In this research, we introduce an automated method to explore a component of natural language processing, using machine learning techniques, to classify online chat room utterances into one of two categories, player or mentor. The proposed automated method relies on a model that emphasizes the use of the UNIGRAMS or tokens in an utterance to decide: players vs mentors. For instance, chat utterances such as; "What should I do?" and "Please check your inbox.", our automatic approach can detect that first chat said by player and second was posted by mentor.

The resulting models and content feature sets (i.e., UNIGRAMS and BIGRAMS) were tested against an intercept-only model (i.e., the baserate) to determine if it was possible to accurately classify utterances according to the role of the speaker (player or mentor).

## 2. METHODOLOGY

### Participants and Procedure

Twenty-one high school-aged participants and two mentors played Urban Science for ten hours over three days as a part of a week-long Conservation Leadership Pro-gram. Players had no prior experience in urban planning and were recruited by out-reach specialists at the Massachusetts Audubon Society's Drumlin Farm Wildlife Sanctuary. Players conversed with their planning team and a human mentor via a chat window. The chat room corpus contained 1963 total utterances, with 972 and 991 utterances from players and mentors, respectively.

### Data Processing and Feature Extraction

Before extracting features, we first addressed the issue of chat-specific terms and emoticons (e.g., "lol", ":)"). These Chat-specific terms and emoticons were treated as individual tokens, as they were intended to convey emotion. Also obvious misspellings (e.g., "Helo") were corrected prior to analysis.

The actual feature vectors were then generated on the basis of this linguistic information by using a "bag of n-grams" approach, i.e. by constructing n-grams (UNIGRAMS and BIGRAMS). In addition to these n-gram counts, we also included punctuation counts, average word length and average utterance length. The part of speech (POS) tagging used the Penn Treebank tagset with some additions specific to the problems related to a chat corpus.

### Automated Classification

We used three automated approaches to classify the utterances of players and mentors, each of which utilizes clas-

Table 1: Automated classifier performance for three approaches based on 10-fold cross-validation experiments. Reported: Accuracy, Precision, Recall and F-measure (baseline ≅ 50.48%).

| | | | Player | | | Mentor | | |
|---|---|---|---|---|---|---|---|---|
| | Features | Acc. | P | R | F | P | R | F |
| CLASSIFIERS | $UNIGRAMS_{svm}$ | **81.03**% | 0.776 | **0.872** | 0.821 | 0.854 | 0.749 | 0.798 |
| | $BIGRAMS_{svm}$ | 76.82% | 0.723 | 0.862 | 0.786 | 0.833 | 0.676 | 0.747 |
| | $UNIGRAMS_{nb}$ | 75.58% | 0.721 | **0.835** | 0.774 | 0.804 | 0.677 | 0.735 |
| | $BIGRAMS_{nb}$ | 75.60% | 0.711 | **0.854** | 0.776 | 0.822 | 0.66 | 0.732 |
| | $UNIGRAMS_{j48}$ | 75.65% | 0.727 | 0.814 | 0.768 | 0.793 | 0.70 | 0.744 |
| | $BIGRAMS_{j48}$ | **60.98**% | 0.562 | 0.961 | 0.709 | 0.874 | **0.265** | 0.407 |

sifiers trained on the dataset. Our classification approach provides us to model both content and context with n-gram features. Specifically, we consider the following two n-gram feature sets, with the corresponding features lowercased and unstemmed: UNIGRAMS and BIGRAMS. Features from the our approaches are used to train Naïve Bayes, Support Vector Machine classifiers, and Decision Tree.

## 3. RESULTS AND DISCUSSION

The accuracy of the induced classifiers were measured using a 10-fold cross-validation method, under its default setting in Weka [1]. The parameters for our model were chosen for each test fold based on standard cross-validation experiments on the training dataset. Table 1 shows the results of the top scores that we managed to achieve with each of the three classifiers. We also used the combination of features and learner parameters that were determined to give the best accuracy by the classifiers. However the results indicated that feature combination had the highest performance. In Table 1, the "Features" column indicates which features were used, and the following columns indicate the results based on Accuracy, Precision, Recall, and F-measure (Acc., P, R, F) for the two roles, player and mentor.

The resulting models from the automated classifiers were quite successful, as each outperformed the 50.48% accuracy attained by the intercept-only model (i.e., selecting 'mentor' for every utterance). However one exception would be the j48 model with BIGRAMS features. Here, we have reduction in performance under the Mentor Recall column in Table 1.

The models based on the SVM method performed the best overall, with 81.03% accuracy on UNIGRAMS and 76.82% accuracy on the BIGRAMS feature set.

In our educational game, the conversation between players and mentor are short, an average three tokens. It proves us that BIGRAMS features performed more significant in short conversations, since BIGRAMS features maybe performs lower due to same features in both classes (however, SVM performs better even in BIGRAMS).

Interestingly, the Naïve Bayes approach performs almost 25% more accurately than baseline (≅50.48%) on both UNIGRAMS and BIGRAMS feature set. It also performed higher than j48, but only on BIGRAMS features.

Overall, all the standard text categorization approach proposed in Section 2 performed between 10% and 31% more accurately than baseline. However, the best performance overall was achieved by SVM on both UNIGRAMS and BIGRAMS features with accuracy of 81.06% and 76.82%, respectively.

Overall, we have presented some preliminary evidence that it is possible to automatically classify individuals' roles in multi-party chat within the context of the serious game Urban Science.

However, it remains to be seen whether these findings will generalize to other serious games. Furthermore, the specific individual roles in this context were preassigned; when peers interact in a collaborative learning environment without predefined roles, certain roles or personalities may arise organically.

Although this presents a significantly more challenging task, it represents a critical step in understanding interaction dynamics in a collaborative learning environment.

## Acknowledgments

This work was funded in part by the Macarthur Foundation, National Science Foundation (REC-0347000, DUE-091934, DRL-0918409, DRL-0946372, BCS 0904909, DRK-12-0918409), the Gates Foundation, and U.S. Department of Home-land Security (Z934002/UTAA08-063).

## 4. REFERENCES

[1] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update; SIGKDD explorations, , volume 11, issue 1. 2009.

[2] F. Keshtkar, C. Burkett, A. Graesser, and H. Li. Detecting players personality behavior with any effort of concealment. In *A. Gelbukh (Ed.): CICLing 2012, Part II, LNCS 7182*, pages 503–515, Springer-Verlag Berlin Heidelberg 2012, 2012.

[3] C. Moldovan, V. Rus, and A. C. Graesser. Automated speech act classification for online chat. In *Procceding of FLAIRS 2011*, 2011.

[4] D. Shaffer, N. Chesler, G. Arastoopour, and DAngelo. Nephrotex: Teaching first year students how to think like engineers. Washington D.C., 2011. Laboratory Improvement (CCLI) PI Conference.

# Categorizing Students' Response Patterns using the Concept of Fractal Dimension

Rasil Warnakulasooriya
Learning Algorithms Team
Pearson
Boston, MA 02116, USA
rasil.warnakulasooriya@pearson.com

William Galen
Learning Algorithms Team
Pearson
Boston, MA 02116, USA
william.galen@pearson.com

## ABSTRACT

We show how students' response patterns can be quantified both globally and locally using the concept of fractal dimension. This metric allows us to identify students who respond to a series of questions and problems in a persistent or anti-persistent manner with implications for personalized just-in-time teaching and learning.

## 1. INTRODUCTION

The motivation behind the present study lies in our observation that students' responses considered as a time series exhibit random walk- or Brownian motion-like characteristics [1]. This observation naturally led to the question of the quantification of such behaviors. Since fundamentally, a random walk-like behavior exhibits irregularities or fluctuations about the expectations, a concept that attempts to look at such irregularities was deemed necessary. One such concept is the fractal dimension [2]. In this paper we show how students' response patterns can be categorized using the concept of fractal dimension, thereby identifying students who do not exhibit persistent response patterns, and hence most likely are struggling with a concept domain.

## 2. THE DATA AND THE METHOD

The response data analyzed in this study originated from a class of about 250 students using MasteringChemistry® (www.masteringchemistry.com) in an introductory chemistry class at a large public university in the United States. MasteringChemistry is an online Socratic homework tutor which allows instructors to assign homework for their students, which are then automatically graded by the platform.[1] The homework problems of tutorial nature in the Mastering system provides students with automated feedback, followup comments, and the opportunity to request declarative and procedural hints at impasses.

The correct (graded 1) or incorrect (graded 0) first attempt responses by students were tracked at the part level (e.g., part A, part B, etc.) of a given online homework question or problem (we will collectively call these items from now on). The "first attempt" was defined as a correct or an incorrect response by a given student to a given part of an item without requesting any hints beforehand. Such interactions were then tracked throughout the semester resulting in about 550 first attempt interactions per student on average, which is the starting point of the retrospective data mining and analysis task that we describe in this paper.

From the first attempt responses described above we derived the "net-score", which is the difference between the number of correct and the number of incorrect first attempt responses at any given instance. The net-score can thus be considered as the *displacement* from the origin for a one dimensional walker with a step to the right considered as a correct response and a step to the left considered as an incorrect response. If the walker is random walking, then the fluctuations in the displacements against the number of steps, and hence the fluctuations in the net-score against the number of first attempt interactions – the response pattern – would be rough or irregular.

### 2.1 Response Patterns & Fractal Dimension

How can we quantify the differences in the response patterns? The concept of fractal dimension can be used to quantify the degree of regularity or roughness of a student's response pattern, which is the variation of the net-score against the number of first attempt interactions (which we will call the net-score space) [3]. Thus, we use fractal dimension as a measure of the roughness of a curve rather than its degree of self-similarity. A student having a perfect net-score (i.e., all correct first attempt responses) would show a straight line in the net-score space, and hence would have a fractal dimension of 1 (this is equally applicable to a student who has all incorrect first attempt responses). In contrast, a student who is randomly responding would show a very irregular pattern, which would ideally cover the 2-dimensional net-score space, and hence would have a fractal dimension of 2. Thus, the fractal dimension values would range from 1 to 2, with lower values corresponding to regular, and higher values corresponding to irregular response patterns. Simulation studies that we have conducted show that it is reasonable to categorize a student as random walking when their fractal dimension reaches a value of 1.8 or above (typical errors are of the order of 0.1).

### 2.2 Global & Local Estimates

The fractal dimension characterization of a student's response pattern can be obtained either globally or locally.

[1]The Mastering$^{TM}$ platform was originally conceived and developed by David E. Pritchard and coworkers at the Massachusetts Institute of Technology, USA. It became part of Pearson's suite of online learning products in 2006.

The former means that we can characterize the response pattern of the entire semester. The latter means that we can characterize subsets of interactions within the semester. We have found that 16% of the students in the class under consideration were random walking globally; that is, these students were responding in a random fashion throughout the semester. The local characterization was done by choosing a first attempt interaction window of length 33 (which roughly corresponds to about 10 items – the typical number of items within an assignment) and shifting this window towards higher interaction values whenever 4 new interactions (which roughly corresponds to the number of first attempt interactions within an item) become available. The time series of a student, with the local fractal dimension estimates and the net-score (i.e., the response pattern) superimposed on the same graph is shown in Figure 1.

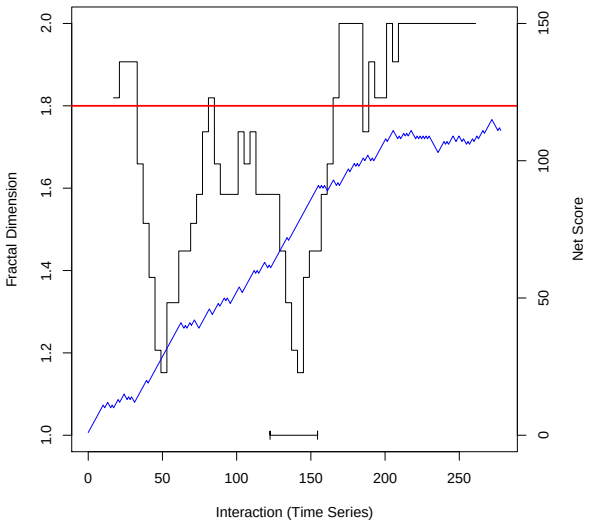



Figure 1: The changes in the fractal dimension and the net-score (blue) along the response pattern for a student having a global fractal dimension of 1.74. Only the first attempt interactions up to the mid-semester are shown. The scale (window of length 33) is also shown.

The particular student shown has a response pattern that can be globally quantified as having a fractal dimension of 1.74, and hence this student cannot be considered as random walking overall throughout the semester. However, the response pattern locally shows interesting features, the most prominent being the onset (around the $150^{th}$ interaction) of an increasing trend in the fractal dimension, and hence, an increasing trend in the irregularity of the response pattern. This shows that the student is struggling around this time. It is worth noting that this difficulty (onset) for the student starts at the beginning of encountering a new concept (stoichiometry), and then showing signs of random response behavior (having a fractal dimension $\geqslant 1.8$) only 5 items later. The concept areas where students act as random walkers can then be clearly distinguished for just-in-time teaching and learning.

# 3. CONCLUSIONS

We have shown that students' response patterns can be quantified using the concept of fractal dimension in a net-score space either locally or globally. We are then able to identify instances where they are in effect responding randomly to a set of items imitating a random walk in one-dimension. It can be questioned whether the net-score or a traditional score alone would not be sufficient to identify students who are struggling. In this context we note that a typical net-score can be reached in many ways leading to different response patterns, and hence, different fractal dimensions. Similarly, a traditional (say, percentage) score provides only a point estimate and also raises issues of what score would indicate mastery. The fractal dimension of a response pattern encodes how a student has achieved that score, which provides more finer-level information than a single score. (The fractal dimension alone would not suffice to identify a struggling student when that student has a consistent but decreasing net-score. The combination of the two would accomplish this task.)

The concept and method we have described and demonstrated above can thus be used as an alert system to identify students at risk and who are struggling within a given concept domain. The method is easily scalable since it only relies on students' response patterns, and hence, a specific student learning model or an instructional model is not needed. Since the method is fundamentally reliant on the responses it is important that the careless errors or lucky guesses are accounted for. Given the complex non-multiple-choice nature of the items with which the students have interacted in this study, the likelihood of lucky guesses can be considered to be negligible. Although the careless error rate may not be as negligible, the correlations that we have investigated with the end-of-term examination show that we are not tapping into noise, and the responses that we have considered are valid. We hope to understand how to correct for careless errors and lucky guesses, and the effect of factoring in the second responses on the fractal dimension of a student's response pattern in the future.

# 4. ACKNOWLEDGEMENTS

We thank Prof. Randall Hall and Prof. Leslie Butler of Louisiana State University, USA, for permission to use their class' data (while preserving its anonymity), and Dr. David Kokorowski for providing us the opportunity to work on this research. We also thank Dr. John Behrens for encouraging us to submit this short paper to the EDM 2012 conference, and Dr. John Stamper for accommodating it.

# 5. REFERENCES

[1] W. Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, New York, 1968.

[2] B. B. Mandelbrot. *Fractal Geometry of Nature*. Freeman, San Francisco, 1982.

[3] T. Gneiting, H. Ševčíková, and D. B. Percival. Estimators of fractal dimension: Assessing the roughness of time series and spatial data. Technical Report 577, University of Washington, Seattle, 2010.

# CurriM: Curriculum Mining

M. Pechenizkiy, N. Trčka, P. De Bra
Department of Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
the Netherlands
{m.pechenizkiy,n.trcka,debra}@tue.nl

Pedro A. Toledo
Department of ISAATC
Universidad de La Laguna
Fac. Fisica. Avda. Francisco Sanchez SN
38203 SC de Tenerife, Spain
pedro@isaatc.ull.es

## ABSTRACT

Curriculum mining includes three main kinds of tasks: (i) *actual* curriculum model discovery, i.e. constructing complete and compact academic curriculum models that are able to reproduce the *observed* behavior of students, (ii) curriculum model conformance checking, i.e. checking whether the observed behavior of students match their expected behavior as defined by the previously discovered or pre-authored curriculum model, and (iii) curriculum model extension, i.e. projecting information extracted from the observed data onto the model, to make the tacit knowledge explicit, facilitate better understanding of the particular academic processes and enable decision making processes. We discuss student and education responsible perspectives on curriculum mining and present the achievements of the ongoing project aiming to develop curriculum mining software including process mining, data mining and visualization techniques.

## 1. INTRODUCTION

An academic curriculum is a (legal) document defining a specific learning program that puts certain types of constraints on how students are expected to take the courses. These constraints typically describe a set of courses and a set of relationships between these courses. The rule *"Algorithms must be passed before Advanced Algorithms can be taken"* can, for example, be seen in a computer science curriculum.

In the current practice however, the rules in a curriculum are usually stated informally, in a natural language, and are thus subject to multiple interpretations. A curriculum is considered simply as the range of courses from which students choose what courses to study. It is not uncommon that students have to approach their study advisors to ask whether they are allowed to enroll in a certain course or what would the impact be if they did. The advisors, on the other hand, are themselves often confused and must ask the board of education for clarification.

To alleviate the above problem we consider formal modeling of a generic academic curriculum. The problem of curricular modeling has been widely recognized as an important an nontrivial task, and so far there have been a few attempts to address it. The problem of authoring and formal modeling of curriculum constraints and personalized study curriculum was addressed in [1]. Modeling an academic curriculum plan as a mixed-initiative constraint satisfaction problem was proposed in [7]. In [5] we identified some typical constraints defined in the existing curriculum and used the formal language of Colored Petri nets (CPNs) to encode these constraints in form of patterns, thus giving a precise and unambiguous semantics to the study rules.

The advantages of having a formal and executable model of a curriculum are not only in its elimination of ambiguity, but in the fact that through the use of CPN Tools and ProM (`http://www.processmining.org`), such a model almost directly offers a wide range of possibilities: students can automatically check, by themselves, whether they are allowed to do something or not; historic data stored in the log of the educational information system can be compared against the model (in this way we could see whether the curriculum was always respected in the past); the same historic data can be used to equip the model with quantitative information (probabilities, delays, etc.) enabling all kinds of performance analysis (finding, e.g., the average time to graduation, the most common paths, etc.) and casting recommendations; and online simulation of the model can facilitate real-time detection of curriculum violations.

In this poster we consider our current ongoing and further planned work on the curriculum mining tools and techniques development. We take both student and education responsible perspectives on curriculum mining that correspond to different information needs.

## 2. CURRICULUM MINING

Figure 1 illustrates the general ideas behind the curriculum modeling and analysis. An information system that supports an educational institution generates event logs that are stored in a database and represent e.g. student performance and enrolment into the courses and corresponding exams. Educators, responsible e.g. for curriculum development and monitoring its effectiveness, can help to identify typical constraints that should be enforced in the study process. Given the event log reflecting historical data and pattern templates we can mine all the patterns present in the database satisfying the templates. The resulting pattern set can be post-processed in a semi-automatic way, and then a unique process model can be assembled into a graph structure. On the process model we can next perform standard task of process mining, like e.g.: (i) determining popular paths and the most important parts in the curriculum, (ii) extending the model with additional information or modifying it, (iii) executing various what if scenarios to facilitate real time decision making, and (iv) real-time monitoring of the process flow as well as monitoring for the changes in the flow or the curriculum.

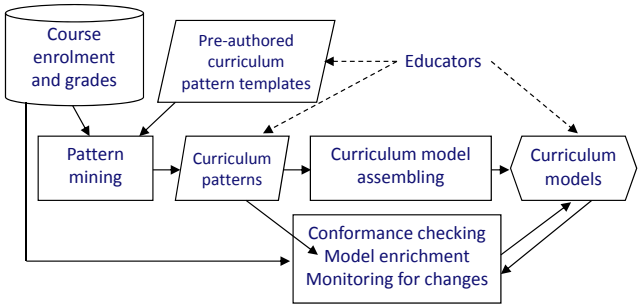



Figure 1: *A conceptual view on the formal modeling of the study curriculum.*

Figure 2 gives an example how the prerequisite constraints and follow up dependencies for the selected or currently recommended key course in the second year are projected to the study curriculum of a particular student. A warning for not having a high enough grade (8) for one of the prerequisite courses as recommended for taking $C$ is highlighted in red.

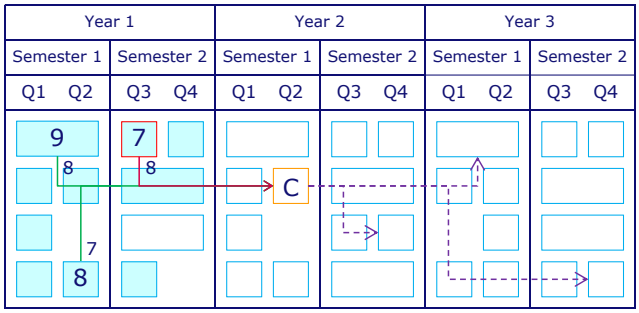



Figure 2: *Projecting patterns on the study curriculum.*

## 3. ONGOING AND FUTURE WORK

Our current efforts are focused on making the *simplest* elements of curriculum mining work in practice such that students, study advisors and directors of education would find them useful. To achieve this we conduct a case study on mining the studying behavior of students at Eindhoven University of Technology over the past 10 years. A typical set of the educators' questions to answer with CurriM include: "What is the *real* academic curriculum (study program)? "How do students *really* study?", "Do current prerequisites make sense?", "Is the particular curriculum constraint obeyed?", "How likely is it that a student will finish the studies successfully or will drop out?. And students' questions like "What are the typical (or the *best*) ways to study?", "What is my expected time to finish?" and "Should I take courses A and B or courses B and D now?".

Besides the relatively straightforward tasks of conformance checking for the known constraints [6], analyzing how much time and effort a particular activity takes [4], and student dropout prediction [3], we work on providing support to answering more elaborate questions. E.g. finding most common types of behavior and clustering them; finding emerging patterns that capture significant differences in the behavior of students who graduated vs. those students who did not or significant changes in behavior of students from one generation and the other; finding frequent patterns that describe a bottleneck in the curriculum, i.e. patterns explaining for which students it is the bottleneck and why. E.g. it is possible to observe that students who take Database modeling theory before passing Logic and set theory or passing it with the minimal grade have to do many more re-examinations for several related courses. We expect that projecting such information to the students will alter their studying behavior, motivating them more strongly to focus on the critical knowledge and skills at the right periods of their studying paths.

It is not uncommon for the curriculum to evolve over time and go through major changes from time to time. Therefore our further work will also include mining and taking into account changes in the study curriculum. This is related to the recently introduced problem of concept drift in process mining [2].

## 4. ACKNOWLEDGEMENTS

This work has been partly funded by Surf Foundation, the Netherlands. We would like to thank STU for providing the data for curriculum mining case study.

## 5. REFERENCES

[1] M. Baldoni, C. Baroglio, I. Brunkhorst, N. Henze, E. Marengo, and V. Patti. Constraint modeling for curriculum planning and validation. *Interactive Learning Environments*, 19(1):81, 123 2011.

[2] R. P. J. C. Bose, W. M. P. van der Aalst, I. Zliobaite, and M. Pechenizkiy. Handling concept drift in process mining. In *Proc. 23rd Int. Conference on Advanced Information Systems Engineering, CAiSE 2011*, volume 6741 of *Lecture Notes in Computer Science*, pages 391–405. Springer.

[3] G. Dekker, M. Pechenizkiy, and J. Vleeshouwers. Predicting students drop out: A case study. In *Proc. 2nd International Conference on Educational Data Mining, EDM 2009*, pages 41–50.

[4] M. Pechenizkiy, N. Trcka, E. Vasilyeva, W. M. P. van der Aalst, and P. D. Bra. Process mining online assessment data. In *Proc. 2nd International Conference on Educational Data Mining, EDM 2009*, pages 279–288, 2009.

[5] N. Trcka and M. Pechenizkiy. From local patterns to global models: Towards domain driven educational process mining. In *Proc. 9th International Conference on Intelligent Systems Design and Applications, ISDA 2009*, pages 1114–1119. IEEE Computer Society.

[6] N. Trcka, M. Pechenizkiy, and W. M. P. van der Aalst. *Handbook of Educational Data Mining*, chapter Process Mining from Educational Data, pages 123–142. CRC Press, 2010.

[7] K. Wu and W. S. Havens. Modelling an academic curriculum plan as a mixed-initiative constraint satisfaction problem. In B. Kégl and G. Lapalme, editors, *Canadian Conference on AI*, volume 3501 of *Lecture Notes in Computer Science*, pages 79–90. Springer, 2005.

# Data mining techniques for design of ITS student models

Ritu Chaturvedi and C. I. Ezeife
School of Computer Science, University of Windsor
Windsor, Canada N9B 3P4
rituch@uwindsor.ca and cezeife@uwindsor.ca

## ABSTRACT

An Intelligent Tutoring System (ITS) is a computer system that provides a direct customized instruction or feedback to students while performing a task in a tutoring system without the intervention of a human. One of the modules of an ITS system is student module which helps to understand the student's learning abilities. Several data mining techniques like association rule mining, clustering and mining using Bayesian networks have been proposed to design effective student models in ITS systems. This paper provides a comparative study of the various data mining techniques and tools that are used in student modeling. We also propose an example-driven approach that can integrate mined concept examples at different difficulty levels with the Bayesian networks in order to influence student learning.

## 1. INTRODUCTION

An ITS system has four main components: interface which provides the means for the students to interact with the ITS through a GUI, expert model which describes the knowledge that represents expertise in the subject matter domain the ITS is teaching, a tutor model that takes or suggests corrective action when necessary and a student model that describes student behavior or knowledge, including his/her misconceptions. Though a lot of work has been done on student model data that describes their cognitive skills like knowledge on a particular concept [2], not too much attention has been paid to outside factors like examples that can be provided as help to students while they use the tutoring system and how they can improve student engagement and their skill level. WebEx [1] is a web-based tool for exploring programming examples that enable teachers to use example-based approach in order to maximize learning opportunity for every student, weak or strong by allowing them to explore program examples in his or her own pace and order. We propose to differ from this approach by rating the examples.

## 2. DATA MINING TECHNIQUES AND TOOLS IN STUDENT MODELING

A correlation between objectives of creating a student model and the mining methods used for the respective objectives is shown in Table 1. If the student model design is directed towards teaching strategies / course planning, where the stakeholder is an educator, the most effective mining methods are clustering and association rule mining, whereas if the student model design is directed towards student learning and improving, then the mining method most effective is classification and prediction. With student as a stakeholder, the most commonly used classification algorithms used are decision tree algorithms like C4.5 and J48 and Bayesian classifiers such as Bayesian networks. To experiment on the comparative analysis of these mining methods, we used two data mining tools: WEKA (http://www.cs.waikato.ac.nz/ml/weka/) and Bayes Server (http://www.bayesserver.com).

| | Objective | Mining method | Algorithms proposed | Most useful for |
|---|---|---|---|---|
| 1 | Measure student's motivation | Clustering | K-means<br>Estimation<br>Mazimization | Teachers |
| 2 | Categorize students as weak or regular based on the hints used, attempts made etc. | | | |
| 3 | Decide if students use resources (hints etc.) and if it has an impact on the marks | Association Rule Mining | Apriori<br>FP-growth | Teachers |
| 4 | Assist students in the sequence they should follow while using the tutor | | | |
| 5 | Find common mistakes | | | |
| 6 | Discover potential student groups with similar characteristics and reactions to a particular pedagogical strategy | Classification Prediction | Decision tree - C4.5, J48<br>Bayesian Classifiers | Student |
| 7 | Detect students' misuse or game-playing | | | |
| 8 | Find common misconceptions that students possess | | | |
| 9 | Identify learners with low motivation | | | |

**Table 1: mining methods used in student model**

### 2.1 Overview of Bayesian Networks

A Bayesian Network is a directed acyclic graph (DAG) of random variables (such as concepts, e.g., Add2Fracs) that uses Bayes theorem to depict probabilistic relationships between these variables. BNs used in student models typically have their DAGs designed by experts and the probabilistic relationships between variables are estimated by using some training data such as that in Table 2. Here, it is assumed that Add2Fracs is probabilistically dependent on Find_LCM and Attempt_made, hence Add2Fracs has these two as its parents.

| Find_LCM | Attempts_made | Add2Fracs |
|---|---|---|
| TRUE | 1 | Pass |
| FALSE | 2 | Fail |
| TRUE | 1 | Fail |
| TRUE | 2 | Pass |
| TRUE | 3 | Pass |
| TRUE | 1 | Fail |
| TRUE | 2 | Pass |
| FALSE | 2 | Fail |
| FALSE | 2 | Pass |
| TRUE | 2 | Pass |

**Table 2: An Example BN Training Dataset**

For a given sample X=(Find_LCM=False, Attempts_made=2), we have to compute $P(X|C_i)$ for each value of class attribute $C_i$ (In table 2, C is Add2Fracs that can have 2 values Pass and Fail) and find the maximum of them. Since $C_i$ has 2 values and X has 2 attributes, the number of such posterior probabilities is 4, as given in the Table 3. Using these probability values, we can compute P(X|Add2Fracs = Pass) = 1/6* 4/6=0.11 and P(X|Add2Fracs=Fail) = 2/4*2/4 = 0.75. This indicates that the probability of failing Add2Fracs is higher for the given sample X. A variable in BN is conditionally independent of all its nondescendants given its parents. The joint probability of the network computed by multiplying the conditional probabilities of each variable given its parents is $P(X_1 \ldots X_N) = \pi_i \ P(X_i \mid Pa(X_i))$ where Pa stands for parents of, i.e., $P(X_i) = P(X_i \mid Pa(X_i))$.

| | |
|---|---|
| P(Find_LCM=False \| Add2Fracs = Pass) | 1/6 |
| P(Find_LCM=False \| Add2Fracs = Fail) | 2/4 |
| P(Attempts_made=2 \| Add2Fracs = Pass | 4/6 |
| P(Attempts_made=2 \| Add2Fracs = Fail) | 2/4 |

**Table 3: Posterior probabilities for Sample X**

When building a student model for the domain of adding fractions, each variable (concepts, tests and other student attributes) is assigned a conditional probability (CPT). Root(s) of the BN will store only prior probabilities in their CPTs. All other variables that are child nodes store conditional probabilities in their respective CPTs. This is shown in Figures 1, 2 and 3.

## 3. PROPOSED ALGORITHM

We are currently working on algorithms to study the effect of relevant examples on student performance. A relational table of examples stores the domain and topic to which they belong, input fraction numbers, detailed solution of the example, its difficulty level and rating. Table 4 shows a sample table.

| Example | Domain | Topic | Frac1_num | Frac1_den | Frac2_num | Frac2_den | Solution | Difficulty | Used | Rating |
|---|---|---|---|---|---|---|---|---|---|---|
| Ex1 | Fractions | Add with CD | 1 | 2 | 1 | 2 | 1/2 + 1/2 = 2/2 = 1 | low | 0 | 1 |
| Ex2 | Fractions | Add with CD | 2 | 3 | 2 | 3 | 2/3 + 2/3 = 4/3 = 1 1/3 | medium | 0 | 1 |
| Ex3 | Fractions | Add with CD | 1 | 4 | 2 | 4 | 1/4 + 2/4 =3/4 | low | 0 | 1 |
| Ex4 | Fractions | Add with DD | 2 | 4 | 3 | 6 | 2/4 + 3/6 = (2*3 + 3*2) / 12 = 12 / 12 =1 | high | 0 | 1 |

**Table 4: Sample example table (CD = Common denominator, DD= Different Denominator)**

When a student uses the ITS for the first time, a new student model is created. This model stores the attributes describing the student and the initial Bayesian network with the respective probabilities for each node. A table of examples is also given as input to the student model. A student is recommended to use examples before attempting a certain activity or task. A comparison of the scores he/she achieves is then made. Thereby, if the student's skill level has increased, the rating of the example is increased. A student is then presented with examples of higher difficulty depending on the skill level he/she has achieved. This method has a two-fold advantage. Examples help students learn a skill better and faster. Secondly, from the logged table of examples, we mine the most useful example (measured by the rating) using an association rule (e.g., Apriori-like or FP-growth like) or sequential pattern algorithm. This information can then be accessed by other student models to help them pick more useful examples. As future work, we propose to use K-means algorithm to cluster examples into different groups so that examples that come before constructing the BN for the student model will ascertain the relationships between variables. Sequential pattern mining can also be applied to the examples to ascertain an effective order in which they are recommended to students. An algorithm is shown below.

Initial BN → Example DB

**1. Student chooses a module (for example Add fractions with Common denominator (eg. current_module=ADD_CD))**
**2. If student wishes to see examples before attempting a task:**
**Until task is achieved or number of example exceeds 3, do**
**- show next example E from current_module**
**- student attempts task from current_module**
**- if task is done successfully,**
**rating of example E is incremented by 1**
**3. If number of example exceeds 3 and the task is still not achieved, then the tutor module is informed (and the student is asked to repeat the module).**

## 4. EXAMPLE STUDENT MODELING

We used WEKA and Bayes Server to describe student attributes and their probabilities. The training dataset D consisted of 100 rows or instances and 10 attributes in a domain of adding fractions. WEKA classified 76% of our instances. There are two steps in creating a Bayesian network: creating DAG and assigning prior and conditional probabilities to each node. These steps can be either done manually by an expert or can be learnt automatically. We preferred to create the DAG manually and input it as an xml file to WEKA but use WEKA to compute the initial probabilities for each node using an input csv file with our dataset D. We used Bayes Server for creating and studying Bayesian networks. Bayes Server does not learn the structure of DAG automatically, but like WEKA, it computes the probabilities from a training data set given as an excel file. Figure 1 shows the DAG generated by Bayes Server for the training set given in table 2. Figure 2 shows the detailed CPT for node Add2Fracs where State1 = False and State2 = True.

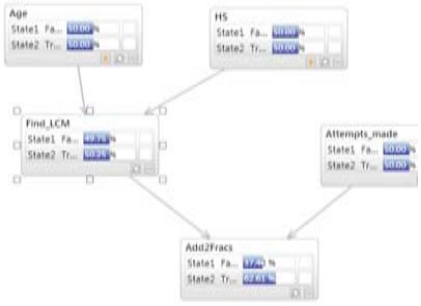



**Figure 1: BN for table 2**

**Figure2: CPT for Add2Fracs**

As evidence is introduced, the CPTs of each node get updated. Figure 3 shows the updated BN when evidence Age = False.

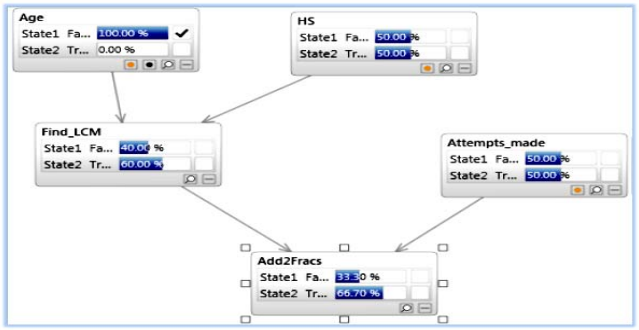



**Figure 3: Updated BN (✓ indicates an evidence)**

## 5. CONCLUSION

ITS Systems have progressed greatly in terms of adjusting instruction, using individualized strategies that are effective for students. However, there is little work done on how aids like examples help students improve their learning skills. This paper attempts to analyze the tools and techniques used in ITS's student models. Future work proposed is to mine up the most important example from several and present it to the student to enhance learning and understanding.

## 6. REFERENCES

[1] Brusilovsky, P. & Yudelson, M. (2008). From WebEx to NavEx: interactive access to annotated program examples. Proceedings of the IEEE, 96, 6, 990–999.

[2] Millán E, Loboda T and Pérez-de-la-Cruz J. 2010. Bayesian networks for student model engineering. Computers & Education 55(4): 1663-1683.

# Deciding on Feedback Polarity and Timing

Stuart Johnson
Alberta Innovates Centre for Machine Learning
University of Alberta
Edmonton, Canada
shjohnso@ualberta.ca

Osmar Zaiane
Alberta Innovates Centre for Machine Learning
University of Alberta
Edmonton, Canada
zaiane@cs.ualberta.ca

## ABSTRACT

This paper outlines the feedback creation and assignment techniques used in a mammography focused Intelligent Tutoring System (ITS) Shufti. Shufti's aim is to provide medical students with an improved learning environment, exposing them to a broad range of examples supported by customized feedback and hints driven by an adaptive Reinforcement Learning system and Clustering techniques.

## 1. INTRODUCTION

Shufti is an Intelligent Tutoring System (ITS) which has been designed to help medical students learn the skills they need to master the complexities of producing a medical diagnosis based on relatively poorly defined, low contrast images. ITS's are tutoring systems which approximate human one-on-one tutoring experiences. Shufti takes the form of a web-based computer game in which learners compete with one another to correctly diagnose images. They are presented with mammograms overlaid with grids and are required to identify Regions Of Interest (ROI) by selecting cells in these grids. As students complete each exercise, they are given a score derived from their accuracy in identifying lesions less points for any hints they may have requested.

Effective human tutors play an active role in the learning experience, providing hints and positive and negative feedback in a strategic fashion. Furthermore, they adapt their feedback to suit the learning styles of their students.

For an ITS to produce similar results it must provide comparable forms of interaction, which is non-trivial for an ITS in the field of mammography as it is an ill-defined domain by the definition of Viger et al [1]. Mammography lacks clear domain models, formal theorems, and cognitive models necessary to automatically teach mammogram diagnosis using conventional ITS construction methods[1]. Consequently, Shufti utilizes a variety of means to effectively simulate attributes of a human tutor.

## 2. APPROACHES TO FEEDBACK

Exercises in Shufti are categorized by difficulty level. Students move from one level to the next after accumulating sufficient points on a certain number of mammograms.

For each exercise Shufti, records the task state transitions which comprise of the exercise state and learner's actions during the exercise. Included in this are current and past states representing their current solution, the last action taken, Shufti's feedback, and the reaction to the feedback by the learner. The *state* is the number of grid cells selected which differ from the exercise solution (i.e. hamming distance). *Actions* are operations such as toggling square selections, certain mouse movements, hint requests, and the submission of an exercise for evaluation. *Reaction* to feedback is whether or not the learner explicitly found the previous feedback helpful.

The polarity of feedback is based on whether it is a positive, encouraging message or a negative, corrective message. The polarity is selected based upon whether the state of the exercise improved or degraded. Degradation or improvement is determined through comparison of current and past hamming distances from the correct state.

Feedback is a critically important part of the effectiveness of a human tutor. To this end, Shufti contains methods for determining the content, polarity and timing of feedback. Shufti employs two feedback control approaches: a clustering-based method and a technique based on Reinforcement Learning.

### 2.1 Clustering-based method

Shufti clusters learners based on their level, points accumulated, the number of requested hints, and the number of exercises they have attempted.

The timing of feedback is governed by a number of different models. *Random* feedback, as its name suggests, occurs randomly. *Timed* feedback is delivered after timed intervals. *After Action* feedback is issued in response to the learner undertaking any action. *Timed After Action* feedback is similar to *After Action* except it is delayed by a specified time. *Random After Action* feedback is similar to *After Action* except it is randomly delivered (it may or may not be issued)

For a given learner, when Shufti has to decide on feedback based on one of the timing models, the potential appreciation of the feedback is assessed based on the task state transitions of similar learners (i.e. learners in the same cluster as the current learner).

If a learner is likely to appreciate a feedback, it is issued. This prediction is based on the likelihood of the reaction to feedback being positive for all available similar records in the task state transition file of all students in the cluster. In the case of a cold start, a random feedback will be issued. This is also performed at random times so as to explore and discover new situations in which feedback may be appropriate.

Clustering allows Shufti to adapt to fit individual learners

or problems. Shufti uses all available data to learn which feedbacks are effective.

## 2.2 RL-based method

Adapting to individual students, though time consuming, is one of the ways in which human tutors offer a superior learning experience. Reinforcement Learning (RL) offers an automated method for an ITS to tune its feedback delivery to individual learners thus approximating a human tutor.

RL is a class of machine learning techniques which resolve problems of mapping situations to actions in order to maximize or minimize a metric[2]. RL allows Shufti to adapt to individual students, learning the most effective times to issue feedback, thus avoiding preset timing models.

An RL system can be thought of as two components; an agent and an environment within which it acts. The environment provides state data and a reward signal to the agent which in turn attempts to maximize the total reward over time. The agent makes use of methods such as Temporal-Difference Learning[2], or Monte Carlo Methods[2] to determine the most long term rewarding action to take in any given state.

Shufti's environment offers task state transitions as state information to the agent. The reward signal is determined by the following formula with the agent seeking to minimize it. It should be noted that when we refer to penalties in the coming paragraph we refer to penalties applied to the agent not to the learner.

$$P = \sigma * count(\tau) + \omega * count(f) - \alpha * score$$

Where $P$ is the total penalty assessed to the agent, $\sigma$ is the penalty assigned over time, $count(\tau)$ is the total time passed, $\omega$ is the feedback penalty, $count(f)$ is the total number of feedbacks given by the agent, $\alpha$ is the reward per score point earned by the user, and *score* is the total score that the user is assigned for the the exercise. Time taken is penalized to encourage the agent to give feedback as a means to hasten the answering of the question. Penalties are also given to the agent each time it gives out a feedback in order to produce strategic feedback selection and timing. In other words, this allows the agent to strike a balance between helping the learner and allowing self driven action. The rate of feedback can be controlled by varying $\sigma$ and $\omega$ with it increasing with $\sigma$ and decreasing as $\omega$ increases. Such variation of the values can either be done automatically so as to simulate the withdrawal of support of a human tutor, or can be done manually by an instructor as part of a larger lesson plan.

This RL-based method not only offers Shufti the ability to control the timing, polarity and content of feedback, but also the ability to adapt to individual learners, thus more closely mimicking human tutors. The downside of this method is the need to understand an effective tutoring strategy for each learner, unlike the previously discussed clustering method which takes advantage of information from many learners in order to adapt.

# 3. COMPETITION

One of the key limitations in traditional training of medical students in imaging analysis is the amount of cases students are exposed to. There are two ways in which Shufti addresses this issue: first of all, Shufti has a very extensive selection of exercises covering a wide range of scenarios unlikely to be seen during a student's short rotation in a radiology department and, secondly, Shufti uses competitive techniques learned from gaming to incent students to expose themselves to as broad a range of scenarios as possible, deepening their knowledge in the field.

Competitive practices in learning have been shown to produce significant improvements in learner performance [3]. To foster competition, Shufti adopts practices from competitive sports and gaming. Learners are not assigned scores based on any one single measure but instead on a composite of measures designed to work with hints from Shufti.

In total, the scores are created based on problem difficulty, answer accuracy, time spent answering the exercise, and learner requested hints. Learners are presented with a wide variety of means to see how they rank next to their peers. In addition to typical public leader boards (commonly used with popular on-line games), Shufti presents performance distribution curves. A learner's overall ranking in Shufti is based on the sum of all scores they have received from all exercises, encouraging them to attempt a large number of exercises.

# 4. HINTS

Hints in Shufti are user-requested, optional pieces of information which aid in solving exercises. They differ from feedback in both how they are issued to the learner and their content. Feedback, for example, takes the form of general statements such as, "Good job!", whereas hints are more direct such as suggesting a general area in which an ROI may be located.

Users are presented with a set of possible hints to request, each hint being labelled with a description of what kind of information the user will receive, along with a specific score penalty which will be applied should the user accept the hint.

Hint penalties ensure the user does not try to improve their score through excessively requesting hints - a phenomenon known as gaming the system [4]. Hint penalties may also have the interesting effect that learners will strategically select the minimum number of hints necessary for them to answer an exercise correctly. Additionally, this causes students to think strategically about which hints they might want to accept, thus broadening their understanding of the diagnostic process.

# 5. REFERENCES


[1] P. Fournier-Viger, R. Nkambou, and E. M. Nguifo. Building intelligent tutoring systems for ill-defined domains. *Studies in Computational Intelligence*, 308, 2010.

[2] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[3] S.-F. Lam, P.-S. Yim, J. S. F. Law, and R. W. Y. Cheung. The effects of competition on achievement motivation in Chinese classrooms. *British Journal of Educational Psychology*, 74(2):281–296, June 2004.

[4] R.S. Baker, A.T. Corbett, K.R. Koedinger, A.Z. Wagner, Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System". ACM Computer-Human Interaction, 383-390, 2004

# Finding Dependent Test Items: An Information Theory Based Approach

Xiaoxun Sun
Australian Council for Educational Research
xiaoxun.sun@gmail.com

## ABSTRACT

In this paper, we propose a new approach to find the most dependent test items in students' response data by adopting the concept of entropy from information theory. We define a distance metric to measures the amount of mutual independency between two items, and it is used to quantify how independent two items are in a test. Based on the proposed measurement, we present a simple yet efficient algorithm to find the best dependency tree from the students' response data, which shows the hierarchical relationship between test items. The extensive experimental study has been performed on synthetic datasets, and results show that the proposed algorithm for finding the best dependency tree is fast and scalable, and the comparison with item correlations has been made to confirm the effectiveness of the approach. Finally, we discuss the possible extension of the method to find dependent item sets and to determine dimensions and sub-dimensions from the data.

## 1. INTRODUCTION

Data mining is the analysis step of the knowledge discovery in databases process, and it is the process of discovering novel and potentially useful information and patterns from large data sets. There are different data mining technologies lying at the intersection of artificial intelligence, machine learning, statistics and database systems. The goal of data mining is to extract useful and previously unknown information out of large complex data collections. Data mining techniques have been applied to many other fields. In the context of educational research, educational data mining refers to developing methods for exploring the unique types of data that come from educational settings, and using existing data mining or developing new methods to better diagnose students' performance and design tests that better suits students.

Students' response data contain the responses of students to a set of test questions. It can be used to determine the knowledge of a student has learned, and it can also be used to discover the relationship between the test items latent or underlying attributes. Such relationship may take the form of attempting to find out which variables are most strongly associated with a single variable of particular interest, or may take the form of attempting to discover which relationships between any two variables are strongest. Students' response data are beneficial to both test developers and course instructors. Students' response data contains valuable information that can be used to improve the effectiveness of test items, and for course instructors, students' performance on the test is importance to instructors for the guidance and improvement of teaching.

## 2. INFORMATION THEORY BASED METHOD

In the information theory, the main concept is entropy. It is defined to measure the expected uncertainty or the amount of information provided by a certain event. We feel more surprised when an unlikely event happens than a likely one occurs. One useful measure of the extent of surprise of an event is to use the logistic function. Suppose the probability of an event happening is $p$, then the extent of surprise of such event can be defined as $-\log_k p$, in which $k$ refers to the base of the logistic function. From this definition, it can be seen that the less the probability is, the higher the amount of information the event would provide. Given the example of students' response data, items that have been answered correctly by a small portion of students contains much more useful information for course instructors than the items that have been answered fully correct.

We adopt the conditional entropy to measure the mutual information, which is a distance metric.

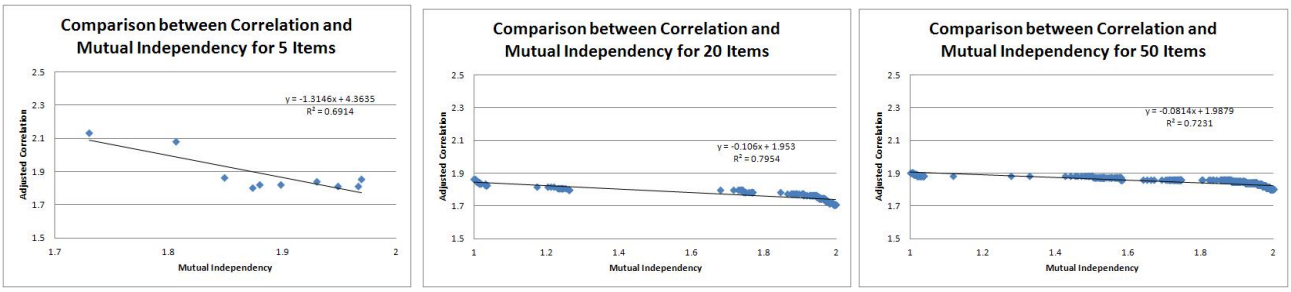
**Definition 1 (Mutual Information Measure).** *The mutual information measure with regard to two random variables $A$ and $B$ is defined as:*

$$MI(A, B) = H(A|B) + H(B|A) \quad (1)$$

Mutual information measure is a measure of how independent are the two random variables when the value of each random variable is known. Two events $A$ and $B$ are independent if and only if their mutual information measure achieves the maximum $H(A) + H(B)$. Therefore, the less the value of the mutual information measure is, the more dependent the two random variables are. According to this measure, $A$ is said to be more dependent on $B$ than $C$, if $MI(A, B) \leq MI(A, C)$.

### 2.1 Finding the Best Dependency Tree

Dependency tree was introduced by Chow and Liu [1] and it has been used in finding dependency structure in the features which improve the classification accuracy of the Bayes network classifiers [3]. [2] used the dependency tree to represent a set of frequent patterns, which can be used to summarize patterns into few profiles. [4] presented a large node




(a) (b) (c)

Figure 1: Comparisons between mutual independency measure and the correlations

dependency tree, in which the nodes are subsets of variables of dataset. The large node dependency tree is applied to density estimation and classification.

## 2.2 Extensions

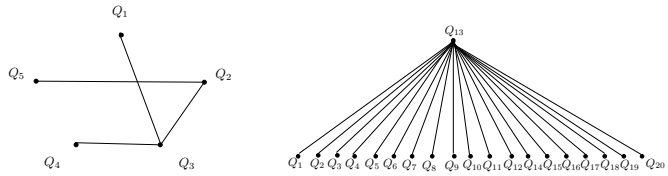
The method described in the paper can be easily extended to be capable of handling two item sets, each of which consists of different items. This extension is useful in the sense that it could provide the dependency relationship in a higher level and the extended method is able to generate the dependency between different sub-strands, which makes entropy-based dependency method superior to the traditional correlation method. The generated best dependency tree could be used to determine dimensions and sub-dimensions of the data. This can be done by summarizing patterns from the best dependency tree.

# 3. SIMULATION RESULTS

In this study, we compare the strength of the correlation against the mutual information measure between two variables. In general, the more independent two variables are, the more related two variables should be. In this sets of experiments, we graph the relationship between mutual independency measure and the correlation of two test items.

Figure 1 shows the relationship between the mutual independency measure and the correlations. Figure 1(a) plots the relationship among items in a short test, Figure 1(b) plots the relationship among items in a medium test, and Figure 1(c) plots the relationship among items in a long test. From Figure 1(a), since 5 items are included in the simulated data, the number of 2-item combinations are $C_5^2 = \frac{5x4}{2} = 10$, for Figure 1(b), 20 items will produce $C_{20}^2 = \frac{20x19}{2} = 190$ different combinations of 2-item sets, and for Figure 1(c), there are $C_{50}^2 = \frac{50x49}{2} = 1225$ combinations for 50 items. From all figures, it can be seen that the slopes of the regression line are negative, which confirms the fact that the more the mutual independency between two variables, the less correlated they are. The $R^2$ of the regression line is the indication of how strong the linear relationship is. In all cases, the values of $R^2$ are greater than 0.65, and in Figure 1(a), the $R^2$ indicates a strong linear relationship, while that relationship is stronger in Figure 1(b) and Figure 1(c).

Figure 2 displays the best dependency tree structure calculated from the simulated data1 and data2. There are two patterns {$Q1$, $Q3$, $Q4$}, {$Q2$, $Q5$}) observed from Figure 2(a) and in Figure 2(b), all the questions are highly dependent on $Q13$.




(a).Best dependency tree constructed from data1

(b).Best dependency tree constructed from data2

Figure 2: The dependency tree structures of two tests

# 4. CONCLUSIONS AND FUTURE WORK

In this paper, we apply the concept of entropy to propose a distance metric to evaluate the amount of mutual information among records in students' response data, and propose a method of constructing dependency tree from the data. The experimental results confirm the effectiveness and efficiency of the proposed method.

There are some potential work on the research agenda. First, the information theory based method presented in this paper finds the dependent item pairs, and it can be extended to calculate the dependency between item sets. Second, the simulation results conducted in this paper are on synthetic data, and applying to real students' response data is necessary.

# 5. REFERENCES

[1] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[2] B. Cui, Y. Li, and Z. Zhang. Summarizing frequent patterns using profiles. *In Database Systems for Advanced Applications, 11th International Conference, DASFAA*, 2006.

[3] N. Friedman, D. Geiger, and M. Goldszmid. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[4] K. Huang, I. King, and M. Lyu. Constructing a large node chow-liu tree based on frequent itemsets. *In Proceedings of the International Conference on Neural Information Processing*, 2002.

# Fit-to-Model Statistics for Evaluating Quality of Bayesian Student Ability Estimation

Ling Tan
Australian Council *for* Educational Research
19 Prospect Hill Rd,
Camberwell, VIC, Australia 3124
ling.tan@acer.edu.au

## ABSTRACT

Bayesian ability estimation is a statistical inferential framework constructed from a measurement model and a prior knowledge model. It is attractive in practice because Bayesian estimation methods offer an elegant way to incorporate appropriate knowledge on target ability distribution in order to improve the accuracy of ability estimation, when there are uncertainties or errors in observable data. One hurdle for applying Bayesian-based methods is evaluating the validity of Bayesian ability estimates at individual-level. This study investigated a class of fit-to-model statistics for quantifying the evidence used in learning Bayesian estimates. The relationship between fit-to-model statistics and root mean square error of Bayesian ability estimation was demonstrated with simulation.

## Keywords

Bayesian ability estimation, student modeling, evaluation.

## 1. INTRODUCTION

Bayesian ability estimation methods have been widely applied in educational testing [2]. Despite its popularity, Bayesian ability estimation is not the standard estimation method of individual ability estimation for several reasons. One criticism of the Bayesian approach is that one can arrive at significantly different answers if different prior distributions are used when analysing the same evidence [2]. In general, two students with the same raw score may get two different Bayesian score if they have different prior distribution. The student having a higher average prior will get a higher Bayesian score than the student having a lower average prior. This criticism invites the methodology of evaluating alternative prior distributions, when more than one prior distribution is available. Another closely related criticism is that Bayesian scores may be biased towards the mean of prior distribution [2]. Again, this points to the necessity of assessing the weight of prior knowledge against evidence from empirical data. This study investigated a class of fit-to-model statistics for quantifying the evidence used in learning Bayesian estimates. The relationship between fit-to-model statistics and root mean square error of Bayesian ability estimation was demonstrated with simulation. In this study, latent ability and evidence are assumed to be uni-dimensional. It means that tasks (or items) come from the same domain, and latent ability can be measured on the same scale as tasks.

## 2. METHODOLOGY

In the context of individual ability estimation, Bayesian ability estimation can be stated as following. Given a student responses $\mathbf{x}=\{x_1, x_2, \ldots, x_L\}$, and a prior ability $\theta$, the posterior distribution of this student ability is written as,

$$P(\theta|\mathbf{x}) = \frac{P(\mathbf{x}|\theta)P(\theta)}{P(\mathbf{x})}$$

$P(\mathbf{x}|\theta)$ represents the causal relationship between latent ability $\theta$ and response vector $\mathbf{x}$. An important assumption in $P(\mathbf{x}|\theta)$ is that no dependence is among item responses given $\theta$. This study uses Expected A Posteriori (EAP) proposed by Bock & Mislevy [1] as Bayesian estimation method. The Expected A Posteriori (EAP) score is the sum of all possible products of $P(\theta|\mathbf{x})$ and $\theta$.

The following section presents a class of fit-to-model statistics for evaluating EAP ability estimation. In this study, the methodology to evaluate evidence is based upon evaluating the conformality of empirical data to an ideal of conjoint measurement. Rasch measurement model is one instance of conjoint measurement. Measurements using this methodology are known as fit-to-model (or model-fit) statistics. These statistics may be based on a residual-based index measuring the distance between observed reponses and the expectation of Rasch-type measurement model. This class of fit-to-model statistics is based on substantial measurement theory. Specifically, these fit-to-model statistics allow one to assess the non-crossing properties of person response functions, which are characterised by $P_i$. When person response functions are parrallel (or non-crossing), the invariance of person order is maintained. In other words, the order of individual abilities is the same across item difficulty scale. In Rasch-type measurement model, both person order and item order are invariant by definition. Therefore, checking the comformality to Rasch model is effectively assessing the quality of evidence.

A simple residual statistic is the squared standardised residual. A mean squared standardised residual is the squared standardised residual divided by the degree of freedom. The mean squared standardised residual fit statistic (MNSQ) [3] for an individual with latent ability estimate $\hat{\theta}$ and observed responses $\mathbf{x}$ of the length $L$ is represented as,

$$MNSQ = \frac{1}{L-1}\sum_{i=1}^{L}\frac{(x_i - E_i)^2}{Var_i}$$

## 3. SIMULATION STUDY

The accuracy of estimated Bayesian abilities was compared with the true abilities. The evaluation of ability estimation is done by using Root Mean Square Error (RMSE), i.e. $RMSE(\hat{\theta}) = \sqrt{\frac{1}{R}\sum_{r=1}^{R}(\hat{\theta}_r - \theta)^2}$, where $\hat{\theta}_r$ is the estimated ability at $r^{th}$ replication, and $R$ is the number of replications. Simulation was performed with the help of ConQuest software [4]. The quadrature points used in marginal probability estimation are 15.

This experiment aims to show the relationship between the MNSQ fit-to-model statistic and the accuracy of Bayesian ability estimates. For each replication, a test of 30 items was generated in normal distribution, $N(0,1)$. A sample of 5,000 response data was generated from a norm ability distribution $N(0,1)$, and the 30-item test. This data set consists of 16% cheating examinees at low ability range (i.e. $\theta \leq -1$), and 16% careless examinees at high ability range $\theta \geq 1$. The cheating responses were created by imputing correct responses to the most difficulty items (i.e. $\delta > 1$), and the careless responses were created by imputing incorrect responses to the easiest items (i.e. $\delta < -1$). Another data set of 5,000 data was generated from the same test without aberrant responses, and this data set was used to set a baseline benchmark. The RMSE was calculated with a replication of 20.

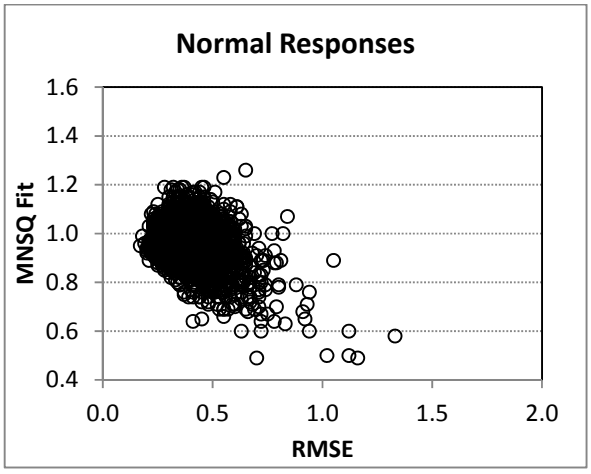



Figure 1a

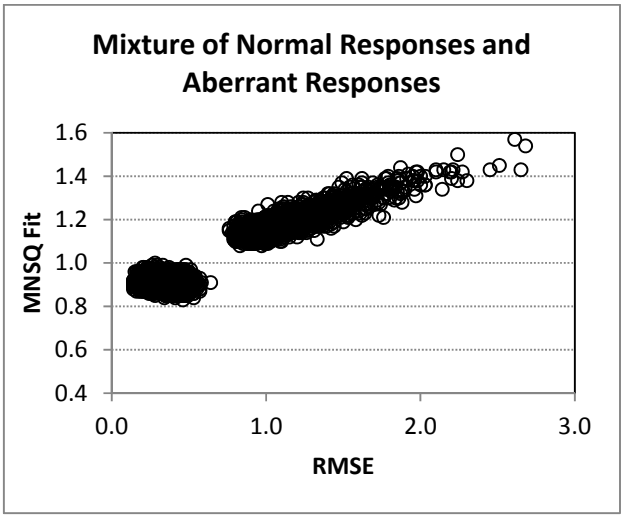



Figure 1b

The Figure 1a shows the relationship between MNSQ fit and RMSE for normal responses, and Figure 1b shows the relationship between MNSQ fit and RMSE for a mixture of normal responses and aberrant responses (i.e. 32%). Both MNSQ and weighted MNSQ fit statistics have an expectation of 1 and variance of $2/L$ [12]. Thus, MNSQ fit values are expected to be centred at 1 and in the range of (0.5, 1.5) for a test with 30 items. MNSQ values less than 0.5 are considered as over-fit, and MNSQ values greater than 1.5 are considered as under-fit. For the normal responses, MNSQs are centred on 1, and they are mostly clustered in the range of (0.6, 1.2), and RMSEs are mostly scattered in the range of (0.2, 0.7). Thus, MNSQ model-fit statistics for normal responses are in a reasonable range. For the mixed data set, responses were scattered in two distinct clusters. The cluster located at the bottom-left has the RMSE in the range of (0.2, 0.65), which is similar to the RMSE in the baseline figure. The bottom-left cluster has the MNSQ in the range of (0.8, 1.0), which is in the range of reasonably good fit. The top-right cluster has the MNSQ in the range of (1.05, 1.4) and the RMSE greater than 0.7. It appears that the MNSQ fit statistic is reasonably sensitive to large RMSE, for at least this experiment.

## REFERENCES


[1] Bock, R.D. & Mislevy, R.J (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement*, 6, 431-444.

[2] Hambleton, R.K. & Swaminathan,H. (1985). *Item response theory principles and applications*. Boston, MA: Kluwer-Nijhoff Publishing.

[3] Wright, B.D. & Stone, M.H. (1979), *Best Test Design*, Chicago, MESA Press.

[4] Wu, M. L., Adams, R. J., Wilson, M. R., Haldane, S.A. (2007). ACER ConQuest Version 2: Generalised item response modelling software [computer program]. Camberwell: Australian Council *for* Educational Research.

# Inferring learners' knowledge from observed actions

Anna N. Rafferty
Computer Science Division
University of California
Berkeley, CA 94720
rafferty@cs.berkeley.edu

Michelle M. LaMar
Graduate School of Education
University of California
Berkeley, CA 94720
mlamar@berkeley.edu

Thomas L. Griffiths
Department of Psychology
University of California
Berkeley, CA 94720
tom_griffiths@berkeley.edu

## ABSTRACT

Teachers gain significant information about their students through close observation of classroom activities. By noting which actions a student takes to achieve particular goals, a teacher can often infer the knowledge possessed by the student and diagnose misconceptions. In this work, we develop a framework for automatically inferring a student's underlying beliefs from a set of observed actions. This framework relies on modeling how student actions follow from beliefs about the effects of those actions. We demonstrate the practicality of this approach by modeling empirical student data from an educational game and validate its performance via a controlled lab experiment. In the educational game, inferences were consistent with conventional assessment measures; in the lab experiment, the model's inferences reflect participants' stated beliefs.

## 1. INTRODUCTION

By observing a student work towards a goal, a teacher can infer what actions the student believes are necessary to achieve the goal and how the student believes those actions affect her progress towards the goal. Critically, a teacher might observe a student's actions and realize that the student has misconceptions about the effects of the actions. This allows for intervention and correction of those specific misconceptions – something that is vital in an educational setting [5]. In this work, we formalize a student model in which the student's knowledge is characterized by her beliefs about how her actions affect the state of the world and what states are most beneficial for achieving her goals. We propose a framework for automatically inferring these beliefs based on observed actions, drawing on ideas from inverse reinforcement learning. In order to make inferences about students' beliefs, we use Markov decision processes to model how those beliefs combine with their goals to determine their actions.

Previous work that has focused on understanding the actions of others has typically assumed that the person taking the actions has full knowledge of how those actions affect the world. For example, work in plan recognition has focused on identifying what the intended plan of action is from some set of already observed actions (e.g., [4]), as well as on categorizing sets of individual actions into strategies and larger semantic parts (e.g., [1]). This work has shown plan recognition to be useful both in human-computer interaction and in understanding data from educational programs. Our approach extends the idea of automated inference about a student's actions to a context in which there may be many actions performed and the person may have misconceptions about how actions affect progress toward the goal.

## 2. INFERRING STUDENT BELIEFS

We consider tasks in which students are trying to achieve some known goal (e.g., win a level in a game) but may have misconceptions about how to achieve that goal. We model these misconceptions using Markov decision processes (MDPs). MDPs provide a natural framework for sequential planning problems in which people must reason about the immediate gains or costs of an action and how that action affects the ease with which the goal can be achieved in the future (see [7] for an overview). An MDP is a tuple $\langle S, A, T, R, \gamma \rangle$, where $S$ is the set of possible states of the world and $A$ is the set of actions that one can take. $T$ represents the transition model $p(s'|s,a)$ specifying the probability of transitioning to a state $s'$ given that the action $a$ was taken in state $s$. $R$ corresponds to the reward model $r(s,a,s')$ that specifies the reward for taking action $a$ in state $s$ and entering state $s'$, while $\gamma$ represents the relative value of immediate versus future rewards. From this specification, one can calculate the expected sum of discounted rewards obtained from each state $s$ and action $a$:

$$Q(s,a) = \sum_{s' \in S} p(s'|a,s) \left( r(s,a,s') + \gamma \sum_{a' \in A} p(a'|s') Q(s',a') \right),$$

which is known as the $Q$-function and can be calculated via a dynamic program [3]. The distribution $p(a|s)$, known as the policy, gives the probability an agent will choose action $a$ given state $s$. As in [2], we model people using a noisily optimal policy in which $p(a|s,T,R,\gamma) \propto \exp(\beta Q(s,a|T,R,\gamma))$, where $\beta$ is a noise parameter.

We assume that students may have misunderstandings about the effects of their actions, and thus their beliefs may not reflect the true transition model. Our goal is to infer what transition model the student believes is correct. Formally, we consider a hypothesis space $\mathcal{T}$ of possible transition models and infer a probability distribution over this space based on the observed student actions. Using Bayes' rule, we can compute the posterior distribution $p(T|\mathbf{a},\mathbf{s}) \propto p(\mathbf{a}|T,\mathbf{s})p(T)$, where $\mathbf{a} = (a_1, \ldots, a_n)$ is the series of observed actions and $\mathbf{s} = (s_1, \ldots, s_n)$ is the corresponding series of states. This posterior distribution represents how likely it is that the student's beliefs correspond to a given hypothesis $T$ by combining the prior $p(T)$ and the likelihood $p(\mathbf{a}|\mathbf{s},T,R,\gamma)$.

The prior can encode knowledge about which misconceptions are common. The likelihood $p(\mathbf{a}|\mathbf{s}, T, R, \gamma)$ represents how well the data fit hypothesis $T$ and can be computed via the Markov property. We can use the posterior distribution over transition models to determine how probable it is that the student has an incorrect understanding of her actions and to calculate what misconceptions are most likely.

# 3. APPLICATIONS

We used the MDP framework to infer learner beliefs in both an educational game and in a learning task in the lab. The first task allowed us to compare the model to traditional assessment measures, while the latter provided an opportunity to more carefully validate the predictions of the model.

## 3.1 Microbe Game

We first applied the MDP framework to data from a publicly available educational game in which students learned about cell biology by playing the part of a microbe navigating through increasingly challenging environments [6]. The student's goal on each of the ten levels is to maximize their chances of surviving the level by purchasing appropriate amounts of mitochondria and chloroplasts. Students may play a level multiple times if they are initially unsuccessful. We modeled data from Level 6 of the game, which introduces sunlight into the environment for the first time. In applying the MDP framework, we consider transition models defined by the number of mitochondria and chloroplasts the student believes are optimal for success. We then infer these beliefs from the series of buying actions and play attempts.

Data came from a pilot study of the educational game conducted in seven schools. A total of 127 students played the game in class or at home and then participated in a post-test to measure content understanding. Post-test scores were analyzed using a standard Rasch Item Response Theory model [8], which yields an ability estimate for each student. The MDP model was used to calculate *maximum a posteriori* (MAP) estimates of students' beliefs about the ideal number of mitochondria and chloroplasts. An analysis of variance on these MAP estimates shows that inferred beliefs were highly significant predictors for estimated ability scores on the post test (mitochondria: $F = 4.9$, $p < 0.001$; chloroplasts: $F = 2.9$, $p < 0.01$). The relationship between average ability estimates of students and their MAP estimates for ideal mitochondria and chloroplasts shows ability peaks at moderate levels of both features. This tracks well with the assumption that the game requires both mitochondria and chloroplasts but excessive amounts waste resources.

## 3.2 Flight Planning Experiment

To validate the MDP framework, we also applied it to modeling learners' actions in a lab experiment where we could collect explicit reports about their beliefs. In this experiment, 25 participants learned to control a spaceship by pressing different buttons. The experiment alternated between phases in which learners could choose what button to press and observe the effect of that action, and flight planning phases in which learners were asked to enter a series of button presses that would move the ship from its current location to another specified location; all participants completed six flight planning phases. Each button press moved the ship by a fixed amount, and learners were told that each button either usually moved the ship in one particular direction or in a direction at random. They could indicate their beliefs about how a button worked using drop-down menus below each button.

We evaluated the model's performance based on how well it matched each participant's stated beliefs in the flight planning phases. Overall, the model achieved relatively high accuracy at inferring learner's beliefs about the buttons that they used: The MAP estimate of the model matched the stated beliefs of the learner in 73% of flight plans. Additionally, in cases where the data were inherently ambiguous such that a human observer would also have difficulty inferring the learner's beliefs, the model tended to place similar posterior mass on all supported hypotheses. This feature suggests the importance of modeling a full posterior distribution rather than only considering the MAP estimate.

# 4. CONCLUSION

We have developed a framework using Markov decision processes for inferring learners' beliefs about the effects of their actions. Such a model has the potential to provide useful feedback to students about their misunderstandings and to provide information to teachers about their students' knowledge. Designing the computational framework for this model is a first step towards applying it in more complex education settings such as virtual labs or games in which more information about the students precise behavior is known.

**Acknowledgements.** We thank WestEd for the use of the microbe game data (supported by NSF grant number DRL-0816359). We also thank Benj Shapiro and HyeYoung Shin for help with data collection. This work was supported by a DoD NDSEG Fellowship to ANR and NSF grant number IIS-0845410 to TLG.

# 5. REFERENCES

[1] O. Amir and Y. Gal. Plan recognition in virtual laboratories. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2392–2397, 2011.

[2] C. L. Baker, R. R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[3] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

[4] H. Kautz and J. F. Allen. Generalized plan recognition. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 32–37. Philadelphia, PA, 1986.

[5] T.-C. Liu, Y.-C. Lin, and Kinshuk. The application of Simulation-Assisted Learning Statistics (SALS) for correcting misconceptions and improving understanding of correlation. *Journal of Computer Assisted Learning*, 26(2):143–158, 2010.

[6] Red Hill Studios. Lifeboat to Mars. `http://www.pbskids.org/lifeboat`, 2009.

[7] R. S. Sutton and A. G. Barto. *Reinforcement learning*. MIT Press, 1998.

[8] M. Wilson. *Constructing measures: An item response modeling approach*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2005.

# Learning Paths in a Non-Personalizing e-Learning Environment[1]

Agathe Merceron, Sebastian Schwarzrock
Beuth Hochschule für Technik Berlin
Amrumer Straße 10
1353 Berlin, Germany
{merceron, sschwarzrock}@beuth-hochschule.de

Margarita Elkina, Andreas Pursian
Hochschule für Wirtschaft und Recht
Alt Friedrichsfelde 60
10315 Berlin, Germany
{margarita.elkina, andreas.pursian}@hwr-berlin.de

Liane Beuster, Albrecht Fortenbacher, Leonard Kappe, Boris Wenzlaff
Hochschule für Technik und Wirtschaft
Wilhelminenhofstraße 75
12459 Berlin, Germany
{liane.beuster, albrecht.fortenbacher, kappe, boris.wenzlaff}@htw-berlin.de

## ABSTRACT

The project LeMo (monitoring of learning processes on personalizing and non-personalizing e-Learning environments) aims to develop a prototype of a web based Educational Data Mining application, which shall provide detailed information on user pattern within e-Learning environments and identify needs of enhancement and revision of the learning offer. The poster presents a case study of analysis of learning paths in a non-personalizing e-Learning environment. Research data have been obtained on the base of log-files during three arbitrarily chosen days.

## Keywords

Learning paths, non personalizing, e-Learning environment.

## 1. INTRODUCTION

The LeMo project [1] is an interdisciplinary research project situated in the field of learning analytics, information science, psychology and data privacy. In order to obtain information about user patterns, as well as about the quality and optimization of e-Learning offers, we integrate in our tool several methods of data mining: association, sequential patterns, regression analysis etc. The tool will be used by eLearning providers, lecturers that use eLearning in different ways, writers of eLearning content and scientists in this field. The main goals for the prototype development are: a data source agnostic back-end, a set of analysis components and a dynamic and adaptive graphical user interface with strong emphasis on an intuitive and easy usability of the application. The prototype focuses on the e-Learning provider role.

## 2. THE TOOL

Being agnostic, the prototype will support different major e-Learning environments rather than a specific one. e-Learning environments can be classical learning management systems like Moodle, where a login personalizing the user is required for access, or online encyclopedias like ChemgaPedia [2] that are non-personalizing environments, where neither login nor registration is needed to access content. To the best of our knowledge, this feature is unique.

Connectors import user data from a specific e-Learning platform into a common data base used for analysis (Fig. 1). Currently two prototype connectors have been implemented, one for Moodle and another one for ChemgaPedia. Connectors for non-personalizing environments or online encyclopedias like ChemgaPedia have to remove fake user data that has been generated by web robots. Currently we have taken a quite cautious approach that might result in suppressing more user data than necessary.

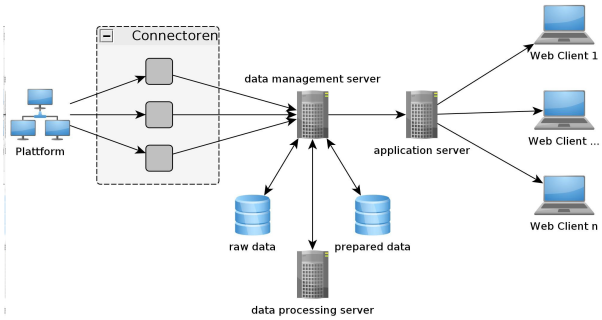



**Figure 1. System architecture**

The methodical guideline for the analysis components and for the adaptive graphical user interface is a catalog of more than eighty questions and research hypotheses collected from our university and business partners. These hypotheses and questions express the information that their authors would like to get from the users' data. The questions of this catalog can be divided into six groups assigned to six topics of analysis: 1. the learning environment, 2. usage of the learning environment, 3. user and groups of users, 4. learning performance, 5. learning paths through the learning environment and 6.communication tools.

Currently we are developing analysis components for topics 2 and 5. In particular we have implemented a component to extract learning paths of learners through the resources of courses. A path is a sequence of resources or learning objects ordered by acces time. Each resource on the path is labeled by its name, a time stamp and a duration. The duration is simply the difference between the time stamp of the present resource and of the following one. The last resource of a path does not have any

[1] This work is partially supported by IFAF and the European Regional Development Fund for the Berlin state.

duration. A resource might appear several times on a path if a user consulted it repeatedly.

## 3. LEARNING PATHS IN AN ONLINE ENCYCLOPEDIA

As a first case study using the path component we have extracted paths of learners in the online encyclopedia ChemgaPedia. ChemgaPedia is a learning environment specialized in chemistry. The content is organized in subject; each subject is subdivided into branches and subareas, each subarea is divided into lessons. It is possible to obtain a lot of information about the usage of ChemgaPedia with GoogleAnalytics: total number of hits, where learners are coming from, on what pages they enter ChemgaPedia, which pages are accessed before leaving and so on. However, due to the high number of pages or resources in ChemgaPedia (more than 18 000), it is not possible, at least not for a non-computer science specialist, to follow paths of users within their session in the encyclopedia. We have extracted all user paths of three days usage data. An overview of the number of paths and their length that have been extracted is given in Table I. For length 10 and higher only samples are given, that should suffice to understand the trend. The second line of table I tells that 154 209 paths of length 1 have been extracted which represents almost 62% of all the extracted paths.

| Length of path | Number of paths | Share |
|---|---|---|
| 1 | 154 209 | 61.68 |
| 2 | 16946 | 6.78 |
| 3 | 4 066 | 1.63 |
| 4 | 2247 | 0.90 |
| 5 | 1404 | 0.57 |
| 6 | 969 | 0.34 |
| 7 | 666 | 0.27 |
| 8 | 478 | 0.19 |
| 9 | 351 | 0.14 |
| 10 | 301 | 0.12 |
| 14 | 86 | 0.03 |
| 17 | 45 | 0.02 |
| 20 | 19 | 0.008 |
| 24 | 14 | 0.007 |
| 27 | 1 | 0.0004 |
| 30 | 1 | 0.0004 |
| 34 | 1 | 0.0004 |

**Table 1. Overview of the number of paths and their lengths**

In this case study, paths are limited to a session within ChemgaPedia. For example, if a user accesses ChemgaPedia, clicks a few pages, follows a link outside ChemgaPedia and comes back to ChemgaPedia again, this will be counted as two different paths. The providers of the encyclopedia already knew through Google Analytics that many learners used the encyclopedia as a reference work, checking some specific information as needed, rather than studying full lessons. Our case study confirms this fact, giving a more precise overview: about 62% of the paths are consulting a single page, and almost 70% of the paths have a length of 6 or less.

Taking longer paths under the microscope leads to the discovering of different recurring navigational patterns: Some users study lessons thoroughly, clicking the pages in the given order and spending about 100 seconds on average on each page, which shows that they study a lesson the way as it has been designed. Other users go back and forth in a lesson, spending 10 seconds on average on a page, rather browsing than studying. This is another kind of information that the providers of ChemgaPedia cannot easily get with Google Analytics.

## 4. CONCLUSION AND FUTURE WORK

We are developing a tool to allow different actors of the educational field to explore how learners are using the e-Learning environment they study with. In a first case study we have extracted the learning paths followed by learners during a session with the freely accessible encyclopedia ChemgaPedia. This study confirms that the encyclopedia is primarily used as a reference. Examining longer paths shows that users are following different navigational patterns. This first result calls for future work in several directions. At first paths have to be presented graphically. Work along these lines is already in progress. Secondly it is not helpful to return numerous individual paths to a teacher, a summary is needed. We plan to investigate frequent sequences and clustering for that purpose. Finally it is also interesting to explore the most common navigational patterns to identify learning styles as proposed in [3].

## 5. ACKNOWLEDGMENTS

We thank our partners for their support and collaboration and especially Kirsten Hantelmann from FIZ Chemie, who enabled our current case study.

## 6. REFERENCES

[1] Beuster, L., Elkina, M., Fortenbacher, A., Kappe, L., Merceron, A., Pursian, A., Schwarzrock, S., Wenzlaff, B.. LeMo- Lernprozessmonitoring auf personalisierenden und nicht personalisierenden Lernplattformen. To appear in: Proceedings of the Conference „Grundfragen Multimedialen Lehrens und Lernens“, Berlin, 2012.

[2] ChemgaPedia homepage, link: http://www.chemgapedia.de

[3] Bousbia, N., Labat, J.-M., Balla, A. and Rebai, I.. Supervised classification on navigational behaviours in web-based learning systems to identify learning styles. International Journal of learning Technology, 6:1 (2011), 24-45.

# Similarity Functions for Collaborative Master Recommendations

Alexandru Surpatean
Department of Knowledge Engineering
Maastricht University
6211 LH, Maastricht, the Netherlands
a.surpatean@student.maastrichtuniversity.nl

Evgueni Smirnov
Department of Knowledge Engineering
Maastricht University
6211 LH, Maastricht, the Netherlands
smirnov@maastrichtuniversity.nl

Nicolai Manie
University College Maastricht
Maastricht University
6211 KH, Maastricht, the Netherlands
nicolai.manie@maastrichtuniversity.nl

## ABSTRACT

A memory-based collaborative system for recommending Master programs has been recently developed for University College Maastricht (UCM). Given the academic profile of a Bachelor student, the system recommends Master programs for that student based on the similarity of her profile to the profiles of the alumni students. The system is operational since September 2011 and is already popular among the UCM students.

This paper considers the question of how to improve the quality of Master recommendations. For that purpose we study several academic profile representations and similarity functions. We identify the best representation strategy and show how to combine recommender systems based on different similarity functions to achieve superior Master recommendations.

## Keywords

Student Similarity, Collaborative Recommender System, Master Program Recommendation

## 1. INTRODUCTION

University College Maastricht (UCM) is a Bachelor program offering a liberal-arts and sciences education. In this study, students can build their own curriculum consisting of approximately 40 out of 157 offered educational modules: courses, skill trainings, and projects. Thus, the academic profiles of the UCM Bachelor students are diverse. To manage the diversity, UCM employs academic advisors, whose task is to help students choose courses in the light of the final goals: desired type of Master programs, jobs, etc.

To facilitate the students and advisors at UCM, we have developed a memory-based collaborative system for recommending Master programs. Given the academic profile (list of past, current, and future academic modules) of a Bachelor student, the system suggests Master programs for that student based on the similarity of her profile to the academic profiles of the alumni students. The tool allows the Bachelor student to modify her own profile, and thus to explore different alternatives in her study and how they influence her Master program possibilities.

This paper considers the question of how to improve the quality of Master recommendations. For that purpose we study two representations of the academic profiles of the students (binary and ECTS[1]-based) and two classes of similarity functions (cosine and Tversky index). We show that: (1) the best representation is ECTS-based and (2) there is no best similarity function. Nevertheless, we introduce an approach to combine recommender systems based on the similarity functions under study so that the resulting combination achieves superior Master recommendations.

[1] European Credit Transfer and Accumulation System.

## 2. MASTER RECOMMENDATIONS

The Master Recommendation problem is given as follows. Let $S$ be the set of all the students, $C$ be the set of all the Bachelor modules, and $M$ be the set of all the Master programs. The *academic profile* of a student $s \in S$ is a vector $p_s$ of values $p_{sc} \in p_s$ corresponding to modules $c \in C$. The values $p_{sc}$ are binary or ECTS-based: if the student $s$ followed or plans to follow module $c$, then $p_{sc}$ equals 1 or the number of ECTS for $c$; otherwise, $p_{sc}$ equals 0. The set of all the academic profiles $p_s$ is denoted by $P$.

In the context of our formalization, the Master Recommendation problem is to find a subset $M' \subseteq M$ of Master programs that fit the academic profile $p_s$ of a student $s \in S$, given data $D \subseteq P \times M$ of academic profiles of alumni Bachelor students, labeled by the Master programs they have chosen. The problem is essentially a classification problem, as each alumni profile is labeled by one Master program, not by a set of programs or preference on them. In this respect, our problem differs from standard recommendation problems where such sets/preferences are available [1].

To solve this Recommendation problem we need a Recommender System $h$: $P \rightarrow 2^M$. We have designed our recommender system $h$ as a memory-based collaborative recommender system [1]. The system memory consists of the training data $D \subseteq P \times M$ of the academic profiles of UCM alumni students labeled by the Master programs they have chosen. The system operates in a collaborative way [1,2]: given the academic profile $p_s$ of a student $s \in S$, the recommender system $h$ returns the set $M'$ of Master programs of the alumni students whose academic profiles are among $k$-closest in the training data $D$ to the profile $p_s$.

To specify completely the recommender system $h$ we need similarity functions over the set $P$ of academic profiles. In this context we note that, for UCM, the set $C$ of modules is much larger than the set of modules a student takes. This implies that the module variables $p_{*c}$ are asymmetric. Thus we need similarity functions for asymmetric binary variables and we choose two such functions: cosine similarity and Tversky index. Given two academic profiles $p_s, p_a \in P$ the functions are defined as follows:

$$\cos(p_s, p_a) = \frac{p_s \bullet p_a}{\|p_s\| \|p_a\|}$$

$$Tversky(p_s, p_a) = \frac{p_s \bullet p_a}{p_s \bullet p_a + \alpha(p_s \bullet \bar{p}_a) + \beta(\bar{p}_s \bullet p_a)}$$

The cosine similarity is a *symmetric* function for asymmetric variables. The Tversky index is an *asymmetric* function for asymmetric variables. If $\alpha=\beta=1$, then it equals the Jaccard distance; if $\alpha>\beta$, we have emphasis on the student to be advised; if $\alpha<\beta$, we have emphasis on the alumni students.

## 3. EXPERIMENTS

We evaluated our recommender system using the leave-one-out method. The UCM data for the system consists of academic profiles of 223 alumni. The total number of Bachelor modules, past and present, to define the academic profiles is 329. The number of unique Master programs to recommend is 147. Among the alumni, 106 followed the same Master as at least one other alumnus. Thus, the academic profiles of these 106 alumni were used in test folds.

Table 1 shows the accuracy of our recommender system. We note that a recommended set of Master programs is correct if the Master program of the student from the test fold is in the set. Since the parameter $k$ increases the size of the recommended set, the accuracy grows with $k$. In addition we note that $k$ is an upper bound on the size of the recommended set of Master programs.

We observe that, in the case of recommendations for Master programs, the similarity functions perform better on average when applied on the ECTS representation, as opposed to the binary representation. Moreover, the prediction accuracy when considering a set of between 2 and 80 recommending neighbors is significantly better when applied on the ECTS representation.

| $k$ | *cos* | | *Tversky* ($\alpha$=1, $\beta$=0) | | *Tversky* ($\alpha$=1, $\beta$=1) Jaccard | | *Tversky* ($\alpha$=0, $\beta$=1) | |
|---|---|---|---|---|---|---|---|---|
| | binary | ECTS | binary | ECTS | binary | ECTS | binary | ECTS |
| 1 | 13.2 | 12.3 | 13.2 | 12.3 | 14.2 | 10.4 | 4.7 | 4.7 |
| 11 | 53.8 | 57.6 | 55.7 | 56.6 | 50.9 | 53.8 | 53.8 | 54.7 |
| 21 | 63.2 | 73.6 | 63.2 | 67.9 | 62.3 | 69.8 | 64.2 | 71.7 |
| 31 | 72.6 | 81.1 | 69.8 | 75.5 | 71.7 | 79.3 | 75.5 | 77.4 |
| 41 | 80.2 | 87.7 | 78.3 | 81.1 | 78.3 | 84.9 | 83.0 | 82.1 |
| 51 | 85.9 | 91.5 | 83.0 | 90.6 | 85.9 | 91.5 | 84.9 | 89.6 |
| 61 | 88.7 | 93.4 | 85.9 | 91.5 | 88.7 | 93.4 | 89.6 | 92.5 |
| 71 | 91.5 | 93.4 | 89.6 | 91.5 | 91.5 | 93.4 | 91.5 | 92.5 |
| 81 | 92.5 | 93.4 | 90.6 | 92.5 | 92.5 | 94.3 | 92.5 | 94.3 |
| 91 | 94.3 | 95.3 | 91.5 | 94.3 | 93.4 | 94.3 | 93.4 | 95.3 |
| 101 | 94.3 | 95.3 | 93.4 | 94.3 | 94.3 | 95.3 | 94.3 | 96.2 |
| 111 | 95.3 | 95.3 | 96.2 | 94.3 | 95.3 | 95.3 | 95.3 | 98.1 |
| 121 | 96.2 | 95.3 | 97.2 | 94.3 | 96.2 | 95.3 | 96.2 | 98.1 |
| 131 | 97.2 | 97.2 | 97.2 | 95.3 | 97.2 | 96.2 | 97.2 | 99.1 |
| 141 | 97.2 | 98.1 | 97.2 | 97.2 | 97.2 | 96.2 | 98.1 | 99.1 |
| 151 | 97.2 | 98.1 | 97.2 | 98.1 | 97.2 | 97.2 | 100 | 99.1 |
| 161 | 98.1 | 99.1 | 98.1 | 98.1 | 98.1 | 98.1 | 100 | 99.1 |
| 171 | 98.1 | 99.1 | 98.1 | 98.1 | 98.1 | 99.1 | 100 | 99.1 |
| 181 | 98.1 | 99.1 | 98.1 | 98.1 | 98.1 | 100 | 100 | 99.1 |
| 191 | 100 | 100 | 98.1 | 99.1 | 100 | 100 | 100 | 100 |

**Table 1. Recommender accuracy versus *k*, as *k* increases.**

Furthermore, we observe that the performance of the classifiers is relatively different in distinct ranges of the $k$ number of neighbors. If we compare the three ECTS-based Tversky indexes in Table 1 we notice that: for $k$ between 1 and 11, Tversky with $\alpha$=1 and $\beta$=0 performs better than the other two; for $k$ between 31 and 81 recommendations, the Jaccard index outperforms the other variants; while for $k$ from 91 to 171, Tversky with $\alpha$=0 and $\beta$=1outperforms the other Tversky variants. We conclude therefore that there is no clear best function when taking a big range for $k$ into consideration.

## 4. COMBINED RECOMMENDING SYSTEM

Figure 1 shows the accuracy curves of two versions $V_1$ and $V_2$ of our recommendation system built on the ECTS-based Tversky similarity functions with $\alpha=\beta=1$ and $\alpha=0$, $\beta=1$. The convex hull of these curves is a set of points that contain the curves. We can build a combined recommender system whose accuracy curve is that of the convex hull. To illustrate the idea assume we need a recommender system whose accuracy curve contains the line segment ($p_1$, $p_2$) in Figure 1. This means that we need a recommender system $V_3$ whose accuracy for some $k$ defines a point $p_3$ on ($p_1$, $p_2$). We design such a system by a very simple approach similar to that from [3]. If we need to determine Master programs for the academic profile $p_s$, we flip a loaded coin with heads probability equal to 1 - *distance*($p_1$, $p_3$) / *distance*($p_1$, $p_2$). If the face-up side is heads, the recommending set of $V_1$ for point $p_1$ is given; otherwise, the recommending set of $V_2$ for point $p_2$ is given. In the long run, it is straightforward to prove that the accuracy of the recommender system $V_3$ for the $k$-value of point $p_3$ will be equal to the accuracy of point $p_3$.

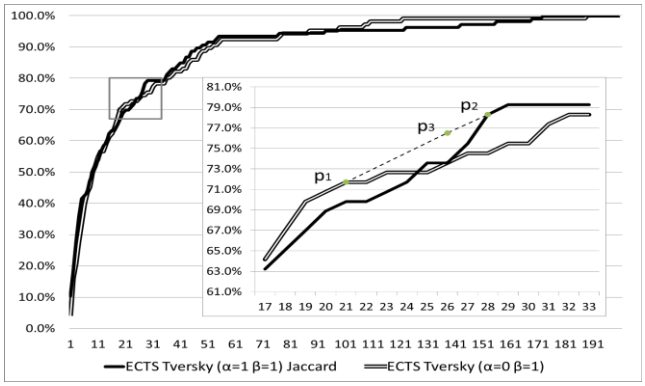



**Figure 1. Accuracy Curves of the Recommender System.**

## 5. CONCLUSION AND FUTURE WORK

This paper showed how to improve the quality of Master recommendations. It determined the best representation of academic profiles and introduced a new approach on how to combine recommender systems based on different similarity functions to achieve superior performance. Future work will focus on implementing and testing this approach.

## 6. ACKNOWLEDGMENTS

We would like to thank the members of the Maastricht University Leading in Learning program for their financial support, as well as Prof. Harm Hospers and the staff of UCM for their help and support.

## 7. REFERENCES

[1] Adomavicius, G. and Tuzhilin, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17* (6). 734-749.

[2] Hastie, T., Tibshirani, R. and Friedman, J. *The elements of statistical learning*. Springer, 2009.

[3] Provost, F. and Fawcett, T. Robust classification systems for imprecise environments. *Proceedings of AAAI/IAAI*, 1998, 706-713.

# Social Networks Analysis for Quantifying Students' Performance in Teamwork

Pedro Crespo
Instituto Superior Técnico
Av. Rovisco Pais 1
1049-001 Lisboa, Portugal
pedrocrespo@ist.utl.pt

Cláudia Antunes
Instituto Superior Técnico
Av. Rovisco Pais 1
1049-001 Lisboa, Portugal
claudia.antunes@ist.utl.pt

## ABSTRACT

Teamwork has become an important part of the educational process, aiming for preparing students for their future performance, but also for making the learning process easier. Predicting students' performance in advance is one of the keys to prevent failure, but teamwork results are difficult to estimate and impair the global prediction results **[1]**.

Working in group is an inherent social activity, and measuring students' performance in it may be enhanced if understood in that context. In this paper, we propose to quantify the performance of students in teamwork by making use of the most effective techniques for social networks analysis. Teamwork is then represented as a network, where students interact with each other, achieving some results (their grades). We explore a network architecture and provide a strategy for quantifying the global contribution of each student through adaptations of the PageRank algorithm **[2]**.

## Keywords

Social networks analysis, Teamwork, Failure prediction.

## 1. Introduction

The concept of Social Networks, typically seen as interactions between individuals, has become extremely popular in the last decade due to its huge application in the online domain in websites such as Facebook, Twitter or LinkedIn. The structure of Social Networks often encloses tremendous amounts of information in the linkage between individuals and content shared among them [3]. Ranking algorithms have already been applied to other domains [4], but seldom applied in the context of education.

In this paper, we show that it is possible to address the thematic of team working in the educational context, through the use of ranking techniques over social networks. Our main goal is to show how these techniques can be applied and what are the main drawbacks faced on trying to measure the value of each student as a team member.

## 2. Teamwork as a Social Network

Teamwork can be defined as social group where students are involved in social interactions with each other, share interests resulting from the terms of classes, and have the common goal of completing a project or assignment where labor can ideally be equally divided among all participants.

When a student agrees to participate in a social group in the context of some subject, three types of grades are achieved in that subject: the final grade, the group grade, and the individual grade.

According to [3], a *social network* is defined as a network of interactions or relationships, where the nodes consist of actors and the edges consist of the relationships or interactions between these actors. A social network is usually represented as a *graph*: a pair $G = (V, E)$ where $V$ is a set of nodes and $E$ a set of edges [5].

Social interactions within students' groups are mutual between all members, so the graph will be *undirected*. In terms of connections we chose to represent *unweighted* edges. Apart from the definition, it is possible to include a *content-based component* containing the students' grades, as seen in Table 1.

**Table 1 – Content-based component: students' grades**

| Student | Final Grade Avg | Group Grade Avg | Individual Grade Avg |
|---|---|---|---|
| 1 | 14 | 16 | 13 |
| 2 | 15 | 16 | 15 |
| 3 | 17 | 17 | 17 |
| 4 | 17 | 18 | 16 |
| 5 | 16 | 18 | 15 |
| 6 | 14 | 17 | 12 |

Figure 1 shows the structure of a social network composed by six students interacting among them. Students 1, 2 and 3 form a social group; students 3, 4 and 5 form a second one, and students 4, 5 and 6 form a third social group. The edges between students 4 and 5 should be seen as a single edge.

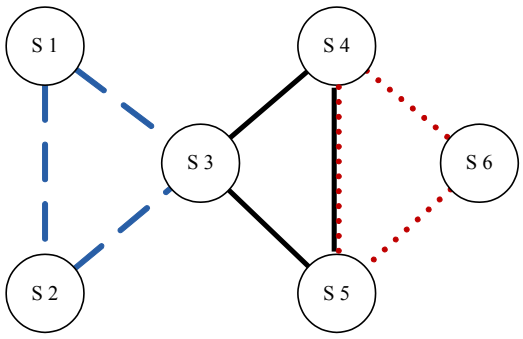



**Figure 1 – Network Structure-based component**

## 3. Link Analysis

In order to evaluate our argument, we conducted some experiments. The data sample contains approximately 1700 evaluations of over 550 unique students. This represents the data of 8 subjects during approximately 2 years combined in 17 evaluation terms. Each student record contains individual grades, group grades and final grades for each enrollment at a given

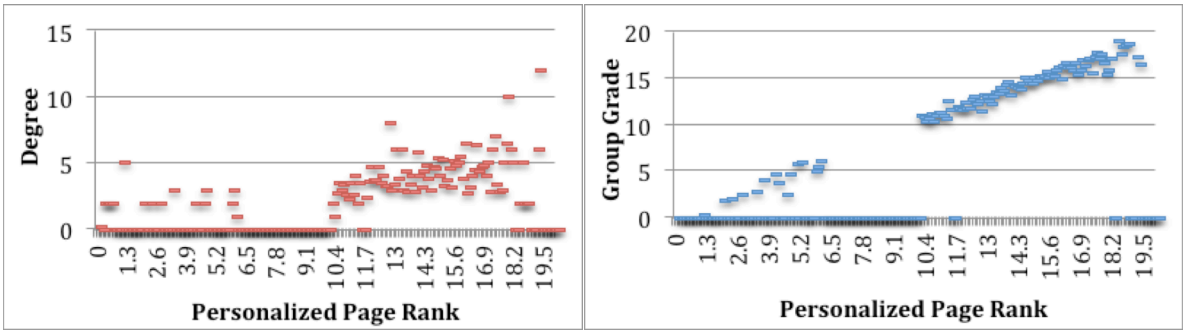



**Figure 2 - Relation between Personalized PageRank and (a) Node Degree and (b) Average Group Grades**

subject. The data was then modeled following the network structure described above.

The best-known algorithms for structure-based analysis fall under the category of ranking algorithms determining the rank of each node in the graph in terms of their similarity.

The PageRank algorithm [2] is known worldwide for ranking web-pages in order of their importance. Given an unweighted graph $G = (V, E)$ as defined above, and where $N$ is the total number of nodes, if an edge exists from a node $j$ to a node $i$ then the node $j$ is diffusing information to node $i$ in terms of importance according to Expression 1:

$$PR(i) = d \sum_{j:j\to i} \frac{PR(j)}{D_{out}(j)} + (1-d)\, v(i)$$

**Expression 1 – PageRank formula**

where $d$ is called the *damping factor* and can take values between 0 and 1. $D_{out}(j)$ represents the out-degree of node $j$. The last component $v(i)$ is part of a personalization vector that can be used to influence the ranking of a given node to better or worse. The Personalized PageRank [6] adaptation is similar to PageRank, differing only in the calculation of $v(i)$. The typical value for $v(i)$ in the Traditional PageRank is $\frac{1}{N}$ but it can vary in the Personalized PageRank.

In the context of this paper we do not see PageRank as a probabilistic distribution, but as the relative value of importance of each node in the graph.

## 3.1 Experimental Results

The PageRank algorithm allows a purely structural analysis based on the representation of a social network. The results from applying PageRank in the described network structure showed a strong proportionality with the degree of each node, and no relation with the group grade.

The Personalized PageRank algorithm allows using content-based data in order to influence the structural analysis by using Personalized PageRank vectors with different content regarding each one of the students. We used the average group grade of each student, present in Table 1 as the value for the PageRank vector but we could have used any other attribute.

Figure 2 (a) shows that the relationship of proportionality between the degree of a node is faded. We can still notice that there is a slight slope towards the highest-ranking values, together with a high dispersion rate. The Personalized PageRank and the average of group grades clearly show a relationship of proportionality in Figure 2 (b).

# 4. Conclusions

The pure structural analysis and the application of the traditional PageRank algorithm fall short from the desired objective of describing the teamwork value of each student. The fact that the ranking simply depends on the degree distribution of the nodes in the graph fails to capture the real value of each student.

Influencing the rankings by adding content-based data through the use of Personalized PageRank vectors seems to have improved the capture of the real teamwork value of each student, but we still need to determine how much of an added value this analysis brings in comparison to simply calculating the group grade average for each student.


# Acknowledgments

This work is supported by Fundação para a Ciência e Tecnologia under research project *educare* (PTDC/EIA-EIA/110058/2009).


# References


[1] Antunes, C. 2010. Anticipating student's failure as soon as possible. In *Handbook for Educational Data Mining*, Romero, C. et al., eds. Chapman & Hall/CRC Press, New York, NY.

[2] Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual Web search engine. In *Proc. of Int'l Conf. World Wide Web* (Brisbane, Australia). Elsevier Science Publishers, 107-117.

[3] Aggarwal, C. 2011. Introduction to Social Network Data Analytics. In *Social Network Data Analytics*, Aggarwal, C., ed. Springer.

[4] Agirre, E. a.S.A. 2009. Personalizing PageRank for World Sense Disambiguation. In *Proc. of EACL '09 Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics* .

[5] Gross, J. L. and Yellen, J. 2004. *Handbook of graph theory*. CRC Press.

[6] Liben-Nowel, D. and Kleinberg, J. 2003. The link prediction problem for social networks. In *Proc. of CIKM '03* .

# Speaking (and touching) to learn: a method for mining the digital footprints of face-to-face collaboration[†]

Roberto Martinez Maldonado, Kalina Yacef, Judy Kay
School of Information Technologies
University of Sydney, NSW 2006, Australia
{roberto, kalina, judy}@it.usyd.edu.au

## ABSTRACT

The research space on educational data mining exploiting data captured from the collaborative learning interactions of students, particularly in face-to-face environments, is vast but still basically unexplored. Students who build a solution in a group have to externalise and make their understandings about the topic explicit to establish common ground with their peers. This offers an enormous opportunity to capture the digital footprints of the process followed by students, these can be used to uncover patterns linked with successful collaboration and learning skills. The full spectrum of emerging technologies to support classroom and small-group work are opening up the possibility to investigate aspects of collocated collaboration. These technologies include interactive tabletops, digital whiteboards and multi-display settings. We present a method to capture, exploit and mine the digital footprints of students working face-to-face to build a concept map at an interactive tabletop. This includes a system that has a mechanism for recording the history of the collaborative process including the partial versions of the solution, applications logs, individual contributions and verbal participation of each student. This paper describes the learning environment, the system to capture a dataset and the data mining techniques that will be used for the study.

## Keywords

Multi-touch tabletop, group modelling, collaborative learning, collocated collaboration, sequence pattern mining

## 1. INTRODUCTION

Students working in small-group activities, in order to collaborate effectively, ought to interact with other participants, who thus need to keep some degree of mutual understanding about the topic under discussion [2]. In collaborative work students have to externalise their points of view and make explicit statements to explain their understanding to others or to regulate the social dynamics. These externalisations include not only verbal explanations but also physical representations according to the tools provided.

Emerging pervasive technologies that support classroom and small-group activities are opening up the possibility to provide novel ways to capture and analyse these externalisations in order to help students collaboration and teachers to orchestrate the classroom [6]. These technologies include shared devices for supporting face-to-face collaboration (interactive tabletops, digital whiteboards, multi-display settings); personal devices that can provide a private workspace and personalised content; and sensors that can monitor aspects of students' participation.

This paper presents a method to capture and exploit the digital footprints of students working face-to-face to build a concept map at an interactive tabletop drawing on research from two main areas: computer-supported collaborative learning and educational data mining. We present a system that has a mechanism for recording a dataset that includes the history of the collaborative interactions that students perform to build the shared solution, partial versions of their final product, applications logs that include the individual contributions and the verbal participation of each student. The data capture is performed in a pervasive manner; thus students can focus all their attention on the activity. This paper describes the learning environment, the apparatus to capture a dataset and the data mining techniques that will be used for the study.

## 2. DESCRIPTION OF THE LEARNING ENVIRONMENT

The system used to sense and capture the learners' face-to-face interactions consists of an augmented interactive tabletop that permits students to discuss and work on the task of building a solution in the form of a concept map at a shared space [3] (Figure 1). The tabletop hardware can detect multiple simultaneous touches. To distinguish between users' touches an overhead depth sensor tracks the position of each user around the table, so recognising which users provided an input. Each single touch performed on the interactive surface is paired with the user. Thus, the system records and logs activity, similarly to any e-learning application. In order to capture the verbal communication of group members, a microphone array that

**Figure. 1. Interactive tabletop learning environment being used to build a joint solution**

[†] http://chai.it.usyd.edu.au/Projects/DataMiningForTabletop

recognises when a learner is speaking is situated above or at one side of the tabletop.

The learning application used to collect the dataset of group interactions uses the well known technique of Concept Mapping [7]. The tabletop application Cmate [4] permits learners to represent their collective understanding about a topic while they discuss and agree on the arrangement and content of the propositions of a group concept map. The learning environment can capture the evolution of the final solution and the individual contributions of each learner to the final product in both, verbal and physical dimensions.

## 2.1 Method

A total of 75 students enrolled mostly in engineering and science courses participated in the study to gather a complete dataset. An initial focus question was posed to the students. The goal for students was to learn, and create a concept map as representation of the Australian Dietary Guidelines 2011 form of concept maps. Participants were grouped in 25 triads. They were initially requested to read an article based on these Guidelines and draw a concept map individually at a personal computer. Then, each group of three students was asked to build a concept map collaboratively at the tabletop. Afterwards, they had to draw an individual concept map again. Pre- and post-tests were conducted as shown in Figure 2. All individual and group actions were logged and recorded from the personal computer application (CmapTools [7]) and the Tabletop environment (Cmate [4]).

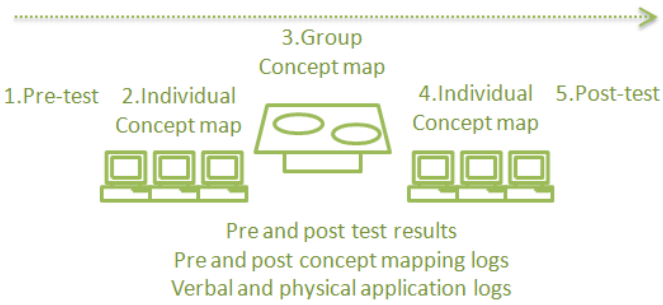



**Figure. 2. Method**

# 3. WORK IN PROGRESS

## 3.1 Dataset challenges

Two key attributes of this tabletop dataset are the sequential order of the actions and the authorship of each. This dataset poses challenges for data mining because the user actions can occur in parallel, be performed by multiple users in a defined order and students can speak while they perform physical actions. We took into account the nature of the data to design data mining objectives to extract frequent patterns of activity and explore which groups favour specific patterns in relation to their performance, nature of collaboration and process followed.

## 3.2 Data mining

One technique that has proven successful in analysing the timing and order of the events is the sequential pattern mining. A sequential pattern is a very frequent consecutive or non-consecutive ordered sub-set of a sequence of events. The data mining objectives for this study are:

*Objective 1: sequence mining by group.* The first approach that can be explored is to mine frequent sequential patterns of interactions and to cluster similar actions to observe whether certain groups favour some strategies used to draw the concept map. This method was introduced by Martinez et. al. [5]. However, that study did not use the verbal participation of students. The verbal actions can either be considered within the sequences, using a proper alphabet, or as a feature present in each sequence or similar sequences.

*Objective 2: sequence mining by student*. This aims to discover the frequent sequences of interactions performed per user at the tabletop. Previous research in group work [1], and more specifically on interactive tabletops [6], found that students behave differently within a group. Some of them work independently, others dominate the activity, under-participate or, in the best of cases, contribute and collaborate equally.

*Objective 3: discovering the building process.* The third objective is to discover and create a visual representation of the process followed by each group to build their final solution. Different strategies can be used to create a concept map. Some groups start by arranging nodes of the graph before creating links. Others start creating links in early stages and others apply a divide and conquer strategy. Different methods for modelling the process using Hidden Markov models or process mining techniques can be used to discover the building process.

## 3.3 Limitations

Current technology limitations forced us to carry out this study in a controlled environment, to assure the quality and consistency of the collected data (e.g. speaker identification and user touch pairing). We observed that our collaborative setting permitted learners to focus on the task, rather than learning particular interaction techniques. A parallel study using the same learning environment is currently being carried out in a real classroom scenario. The present study does not include speech recognition.

# 5. REFERENCES

[1] Anaya, A. and Boticario, J. Clustering learners according to their collaboration. *Computer Supported Cooperative Work in Design* IEEE (2009), 540-545.

[2] Dillenbourg, P. What do you mean by 'collaborative learning'? *Collaborative Learning: Cognitive and Computational Approaches. Advances in Learning and Instruction Series.* Elsevier Science (1998), 1-19.

[3] Martinez, R., Collins, A., Kay, J. and Yacef, K. Who did what? who said that? Collaid: an environment for capturing traces of collaborative learning at the tabletop. *Interactive Tabletops and Surfaces* ACM (2011).

[4] Martinez, R., Kay, J. and Yacef, K. Visualisations for longitudinal participation, contribution and progress of a collaborative task at the tabletop. *CSCL* (2011), 25-32.

[5] Martinez, R., Yacef, K., Kay, J., Kharrufa, A. and Al-Qaraghuli, A. Analysing frequent sequential patterns of collaborative learning activity around an interactive tabletop. *4th EDM* (2011), 111-120.

[6] Martinez, R., Yacef, K., Kay, J. and Schwendimann, B. Unpacking traces of collaboration from multimodal data of collaborative concept mapping at a tabletop. ICLS (2012).

[7] Novak, J. and Cañas, A. *The Theory Underlying Concept Maps and How to Construct and Use Them* Florida Institute for Human and Machine Cognition, 2008.

# Stress Analytics in Education

Rafal Kocielnik, Mykola Pechenizkiy, Natalia Sidorova
Department of Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
the Netherlands
{r.d.kocielnik, m.pechenizkiy, n.sidorova}@tue.nl

## ABSTRACT

During the years of college and university education students are exposed to different kinds of stress, especially during the difficult studying periods like final exams weeks or project deadlines. Stress on a long run is dangerous and can contribute to illness through its physiological effects or maladaptive health behaviors. Many students admit, or are self-aware, that they become stressed under different circumstances and have some clues about their potential stressor. Still, even for such students, the monitoring and awareness of stress are not systematic and based on subjective data, i.e. someone's feelings. In our work we aim at providing means to students to become aware of the past, current and expected (objectively measured) stress and its correlation with their performance, to understand their stressors, to cope with and prevent stress - thus, to live healthier and happier lives and better organize their studies.

## 1. INTRODUCTION

Stress has become a serious problem affecting many people of different professions, life situations, and age groups. The workplace has changed dramatically due to globalization of the economy, use of new information and communications technologies, growing diversity in the workplace, and increased mental workload. In the 2000 European Working Conditions Survey (EWCS)[1], work-related stress was found to be the second most common work-related health problem across the EU. Similar emphasize can be observed from the surveys in US. Although not studied thoroughly (yet), similar threats has been observed in education. According to a survey by fifteen university and college newspapers in the Netherlands[2] many students themselves confirm that they encounter serious stress in related to their studies. E.g. in Delft University of Technology, more than 450 students filled out the questionnaire and 52.9% of them said 'yes' to the question: *"Have you ever experienced huge stress on your studies?"*. Moreover, stress might not even be observed as problematic by the persons themselves; high levels of stress are often perceived by people as a norm, a signal that they do their best to achieve their goals.

Stress can contribute to illness directly, through its physiological effects, or indirectly, through maladaptive health behaviors (smoking, poor eating habits, lack of sleep, etc.) [4]. It is important to motivate people to adjust their life style or start using appropriate stress coping strategies.

Being enrolled in a study program at a college or university students may encounter a number of challenges[3]. There is a continuous pressure (whether coming from parents, increasing competition on a job market, society in general, or other students) to perform well. Students regularly encounter difficult studying periods like final exams weeks or (group) project deadlines. Many students may experience financial insecurity, problems with communicating to other students, loneliness. Becoming independent, many students develop bad habits like smoking, drinking, taking drugs, Internet addiction, and gambling.

In general, there are a number of factors that are likely to cause stress for students including but not limited to long study hours, work overload, time pressure, difficult, demanding or complex tasks, high responsibility, lack of breaks, conflicts, lack of training, income insecurity, poor physical work conditions (limited space, inconvenient temperature, limited or inappropriate lighting conditions) as many of such factors have been found to be important in the analysis of stress at work [5].

The complete avoidance of stress in education is impossible. Still, if people are *informed of their stress levels*, they become empowered for taking some preemptive actions in order to alleviate stress.When educational organizations become aware of stress-related problems they figure out different ways to help students. It is becoming a more popular practice to promote healthier lifestyle to students, give advices regarding safety and security, eating well, anger management, dealing with emotional abuse or a culture shock, overcoming loneliness, and alike. However, still not that much is done to systematically monitor the levels of stress that students experience. In the current practice students often get help only when the problem becomes severe and may require psychological or medical help.

Appearance of modern sensor technologies does allow measure stress level based on objective physiological signals in non-lab settings, i.e. in a daily life. Our goal is to provide a *stress analytics* solution that would help students to monitor, understand and manage their stress thus preventing severe stress-related problems.

[1] http://www.eurofound.europa.eu/pubdocs/2001/21/en/1/ef0121en.pdf

[2] http://web.tue.nl/cursor/internet/jaargang54/cursor12/nieuws/index.php?page=x37

[3] http://www.onlineeducation.net/resources/stress-facing-students

## 2. APPROACH – STRESS ANALYTICS

In [1] we proposed the conceptual framework for managing stress at work. One very important step in the process of stress management is making the person aware of the past, current and expected stress. We automate the identification of the stress from sensor data and the common stressors from the other data sources and meta-data that a student is willing to provide to the system, including e.g. the schedule, calendar events and daily routing of the student, and personal correspondence in e-mail and social media.

Figure 1 illustrates the main objective of the stress analytics - to provide an overview of the stressors, allow to explore them interactively with OLAP style operations on the data cube and to enable access to the evidence, i.e. (annotated) physiological signals (Figure 2, top).

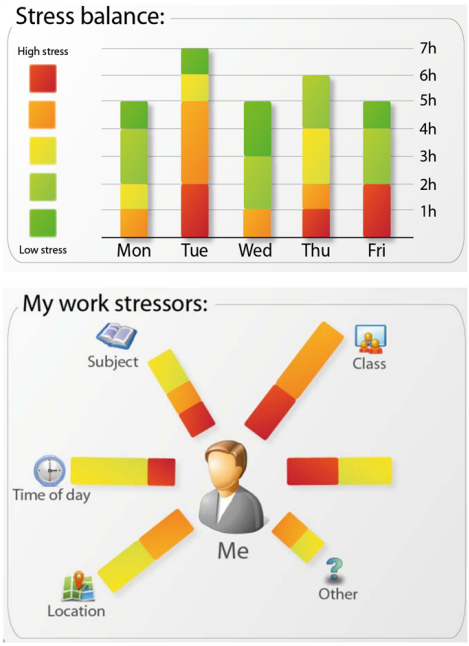



Figure 1: *Interactive overview of stressors*

To achieve this we measure stress-related physiological signs from the sensor data. Galvanic skin response (GSR) reflecting sweating is known to be highly correlated with the level of stress a person experiences [3]. In [2] we proposed an automated approach for the acute stress detection from GSR signal. We make stress and stressors visible by keeping track of the calendar events and daily routine of the worker (Figure 2, bottom), and relating these events with detected stress from the sensor data. We also enable automated analysis of additional information sources, such as sentiment classification of the incoming and outgoing e-mails or social media messages[4] and explicit user feedback. Besides being an awareness tool, stress analytics supports data mining on the already aligned data. By discovering relationships between event data and stress occurrences we can further help a student to focus on the most important patterns observed in the past and cast predictions regarding the foreseeing stressful events in her agenda.

[4] `www.win.tue.nl/~mpechen/projects/senticorr/`

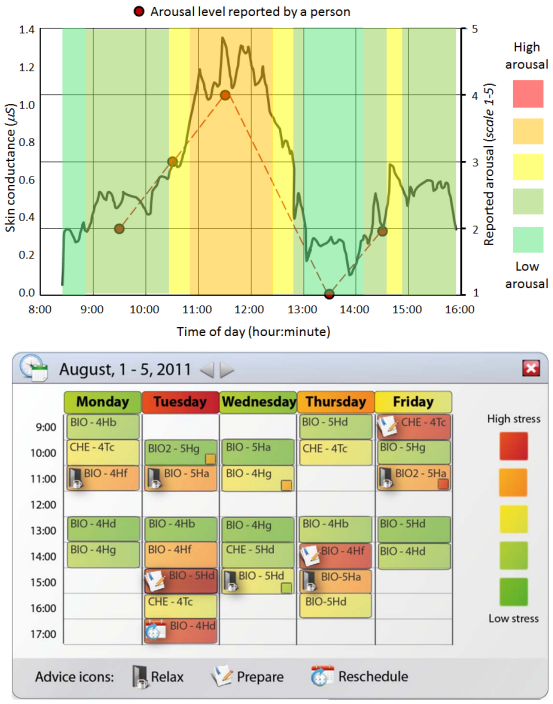



Figure 2: *Stress levels: Monday (top) and the week (bottom)*

## 3. ONGOING WORK

Our current efforts are focused on two kinds of tasks. From the technology perspective we develop the infrastructure for stress-related data management and processing. This include raw and preprocessed multi-modal data storage and indexing, data schema for efficient data cube operations, data mining techniques supporting feature extraction, stress detection, pattern mining, classification and prediction, and visual interactive data exploration. From the domain perspective we perform different case studies, collecting subjective and measuring objective stress before, during and after the group project presentations, partial exams as online multiple choice tests and regular written and oral examinations.

## 4. REFERENCES


[1] J. Bakker, L. Holenderski, R. Kocielnik, M. Pechenizkiy, and N. Sidorova. Stress@work: From measuring stress to its understanding, prediction and handling with personalized coaching. In *2nd ACM IHI 2012*, pages 673–678.

[2] J. Bakker, M. Pechenizkiy, and N. Sidorova. What's your current stress level? Detection of stress patterns from GSR sensor data. In *IEEE ICDM 2011 Workshops*, pages 1247–1250.

[3] W. Boucsein. *Electrodermal activity*. New York and London: Plenum Press, 1992.

[4] K. Glanz and M. Schwartz. Stress, coping, and health behavior. *Health behavior and health education: Theory, research, and practice*, pages 211–236, 2008.

[5] S. Michie. Causes and management of stress at work. *Occupational and Environmental Medicine*, 59(1):67, 2002.

# Variable Construction and Causal Discovery for Cognitive Tutor Log Data: Initial Results

Stephen E. Fancsali
Department of Philosophy
Carnegie Mellon University
& Apollo Group, Inc.

sfancsali@gmail.com

## ABSTRACT

We present a method to simultaneously search for student-level variables constructed from Cognitive Tutor log data and graphical causal models. We seek causal explanations of behavior in Cognitive Tutors, including "gaming the system" and off-task behavior, selecting variables by their contribution to causal structure and strength learning.

## Keywords

causality, intelligent tutors, discovery with models

## 1. INTRODUCTION[1]

Researchers constructing student-level statistical and causal models from "raw" log data of courseware must construct variables to represent student-level, aggregate features of interest. We propose search for constructed variables, with a focus on "gaming the system" and off-task behavior in Cognitive Tutors. Variables are assessed by their support of inferences about causation and causal strength in graphical causal models [12].

"Feature engineering" has been explored for predictive models of educational data (e.g., [1], [4]). Usually targets (not always, cf. [6]) have been fine-grained outcomes (e.g., success at next tutor interaction) and not student-level outcomes (e.g., exam scores).

Graphical causal models have also been used on educational data ([10], [11], etc.) with student-level features. This work develops an approach [9] that combines data-driven variable construction and algorithmic causal discovery to model student behavior.

## 2. DATA + MOTIVATION

Data are from interactions of 102 non-traditional, adult learners with the Carnegie Learning Algebra Cognitive Tutor in an (online or on-campus) algebra course at the University of Phoenix, specifically data from the last module of the course. Target learning outcomes are students' course final exam scores.

Learners "game the system" by taking advantage of intelligent tutor properties to get through course material without genuinely learning [2]. Off-task learners disengage from the tutor and behave in ways unrelated to learning tasks [7]. Both types of behavior have been associated with decreased learning [3].

Research on gaming describes it as "harmful" in a non-causal way, denoting mere association with negative outcomes. Methods for inferring causal relationships from observational data may help provide evidence for (or against) a causal relationship between gaming behavior and learning.

We deploy software "detectors" of gaming [5] and off-task behavior [7] that use a variety of "engineered"/"distilled" features [4] to determine whether a transaction corresponds to gaming or off-task behavior. We treat their output as "fine-grained" observations of behavior and seek variables to represent aggregate behavior over "fine-grained" observations.

Recent work [8] considers whether influences of gaming the system and off-task behavior on learning are immediate, aggregate or both. They construct variables over lessons/units of interest and report that gaming is weakly associated with aggregate poorer learning and that off-task behavior is strongly associated with aggregate poorer learning.

## 3. METHOD + RESULTS

Our data range over several units (with corresponding sections) of algebra material and 32 skills. Student behavior in particular units, sections, or skills is possibly more important for learning than behavior over the entire module. We search over variables constructed at these levels of aggregation. Models in [8] consider variables as *counts* of gamed/off-task steps. As they suggest, other functions might manifest important behaviors. Our strategy is to search over constructed variables to find those that support causal inferences using algorithmic search for causal models.

Graphical causal models are frequently directed acyclic graphs (DAGs) with associated probability distributions (Bayesian networks). DAG nodes represent variables; edges represent causal relationships. Two assumptions[2] link causal structure represented by a DAG to independencies entailed by a DAG: the Causal Markov Condition and Causal Faithfulness Condition [12].

Algorithms like FCI[3] [12] learn the equivalence class of graphs compatible with conditional independence relations among measured variables, assuming there may be unmeasured common causes of measured variables. FCI returns a partial ancestral graph (PAG), representing the set of causal graphs compatible with conditional independence relations among measured variables. Edge interpretations in a PAG are $X$ o→ $Y$: Either $X$ causes $Y$, or $X$ and $Y$ share a latent common cause (or both); $X$ o—o $Y$: (1) $X$ causes $Y$, (2) $Y$ causes $X$, (3) $X$ and $Y$ share a latent common cause, or (4) either (1) & (3) or (2) & (3); $X \leftrightarrow Y$: There is a latent common cause of $X$ and $Y$; $X \rightarrow Y$: $X$ causes $Y$.

To judge which variables support causal inferences with uncertain causal structure, we iterate[4] over DAGs consistent with a PAG

[1] The author gratefully acknowledges the Apollo Group, Inc., for funding as well as Ryan Baker, David Danks, Sujith Gowda, Tristan Nixon, Joseph Ramsey, Steve Ritter, Partha Saha, and Richard Scheines for assistance and advice.

[2] We omit illustrations of the framework and assumptions.

[3] The Tetrad Project <http://www.phil.cmu.edu/projects/tetrad/>

[4] Our heuristic is not ideal; a better method is work in progress.

over sets of constructed variables and calculate "average causal predictability" achieved. For each compatible DAG, we specify a linear regression model for *final_exam* score with its direct causes as predictors. If each DAG is equi-probable, we maximize average $R^2$ value over these models.

Fig. 1 is a baseline causal model (PAG) for variables from [8]. The negative association of *number_steps_gamed* and *final_exam* is likely induced by a causal relationship. Average causal $R^2$ ("causal predictability") for this PAG is .5028.

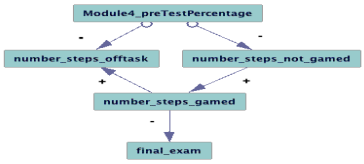



**Fig. 1. Baseline PAG; +/- indicate association**

We search over aggregate variables constructed from characteristics tracked by Cognitive Tutors or calculated by "detectors" – counts (and when relevant, proportions) of transactions: overall, correct, wrong, help/hint request, "known bugs" (misconceptions), gamed, off-task; transaction time taken; average gaming & off-task (numerical) estimate; counts (and relevant proportions) of: steps, gamed steps, and off-task steps. We consider aggregating over the entire module and sections, units, and skills within the module. For each level, we consider functions of step-level characteristics (and their natural logarithm) to determine constructed variables: sum, average, variance, max, and min. The schema for constructed variable names is: *LEVEL_level-name_function(characteristic)*.

Applying functions at different aggregation levels, we "explode" a set of a few hundred variables and "prune" by removing uninformative and redundant variables; for highly correlated pairs, we remove the variable with lower target correlation. From 20 variables with highest correlation to the target, we randomly select sets of (9) variables, apply FCI, and seek the set that maximizes average causal $R^2$ afforded by the PAG. More work is required to determine the best sizes for these variable sets.

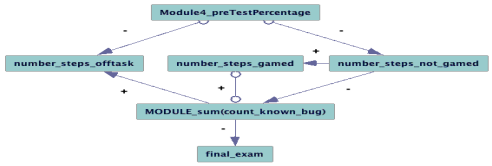



**Fig. 2. Augmented baseline PAG; avg. causal $R^2$ = .5816**

Full search shows the importance of the count of misconceptions in the module: *MODULE_sum(count_known_bugs)*. Augmenting our baseline variables with this, we apply FCI (PAG, Fig. 2). Our baseline model suggests the o—o edge between *number_steps_gamed* and "misconceptions" can be oriented as →; the latter is a more proximate cause of learning.

For our full search, our training set consists of variables computed over 80% of steps randomly sampled over all students. Test set variables are computed over the remaining steps. The PAG (Fig. 3) that maximizes causal predictability for the training set has average causal $R^2$ = .614 (test set avg. causal $R^2$ = .5137).

We establish that at least one variable mediates a (likely) causal link between "gaming" and learning. Future work will extend and generalize this approach and refine the search space.

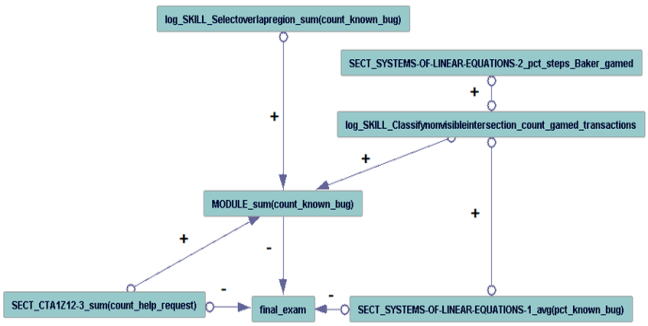



**Fig. 3. PAG for full variable search**

## 4. REFERENCES


[1] Arnold, A., Beck, J., Scheines, R. Feature Discovery in the Context of Educational Data Mining: An Inductive Approach. Proceedings of the AAAI 2006 Workshop on Educational Data Mining, (2006), 7-13.

[2] Baker, R.S.J.d., Corbett, A.T., Koedinger, K.R., Evenson, E., Roll, I., Wagner, A.Z., Naim, M., Raspat, J., Baker, D.J., Beck, J. Adapting to When Students Game an Intelligent Tutoring System. Proceedings of the 8th International Conference on Intelligent Tutoring Systems, (2006), 392-401.

[3] Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System." Proceedings of ACM CHI 2004, 383-390.

[4] Baker, R.S.J.d., Corbett, A.T., Roll, I., Koedinger, K.R. Developing a Generalizable Detector of When Students Game the System. User Modeling and User-Adapted Interaction, 18 (2008), 287-314.

[5] Baker, R.S.J.d., de Carvalho, A. M. J. A. Labeling Student Behavior Faster and More Precisely with Text Replays. Proceedings of the 1st International Conference on Educational Data Mining. (2008), 38-47.

[6] Baker, R.S.J.d., Gowda, S.M., Corbett, A.T. Automatically Detecting a Student's Preparation for Future Learning: Help Use is Key. Proceedings of the 4th International Conference on Educational Data Mining, (2011), 179-188.

[7] Baker, R.S.J.d. Modeling and Understanding Students' Off-Task Behavior in Intelligent Tutoring Systems. Proceedings of the 2007 Conference on Human Factors in Computing Systems, (2007), 1059-1068.

[8] Cocea, M., Hershkovitz, A., Baker, R.S.J.d. The Impact of Off-Task and Gaming Behavior on Learning: Immediate or Aggregate? Proceedings of the 14th International Conference on Artificial Intelligence in Education, (2009), 507-514.

[9] Fancsali, S.E. Variable Construction and Causal Modeling of Online Education Messaging Data: Initial Results. Proceedings of the 4th International Conference on Educational Data Mining, (2011), 331-332.

[10] Rai, D., Beck, J.E. Exploring User Data from a Game-Like Math Tutor: A Case Study in Causal Modeling. Proceedings of the 4th International Conference on Educational Data Mining, (2011), 307-311.

[11] Scheines, R., Leinhardt G., Smith, J., Cho, K. Replacing Lecture with Web-Based Course Materials. Journal of Educational Computing Research, 32 (2005), 1-26.

[12] Spirtes, P., Glymour C., Scheines, R. Causation, Prediction, and Search. 2nd Edition. MIT, Cambridge MA, 2000.

# List of authors


**A**

Adarsh, G V 10
Aleven, Vincent 73, 126
Antunes, Claudia 232
Aros, Carolina 210
Azevedo, Roger 65

**B**

Baker, Ryan S.J.D. 126, 152
Baldwin, Julia 160
Barker-Plummer, Dave 134
Barnes, Tiffany 164
Bayer, Jaroslav 103
Beck, Joseph 200
Beheshti, Behzad 81
Bergner, Yoav 95
Beuster, Liane 228
Biswas, Gautam 57, 65
Bouchet, François 65
Brunskill, Emma 118, 188
Butler, Heather 172
Bydzovska, Hana 103

**C**

Cai, Zhiqiang 172
Campagni, Renza 208
Chaturvedi, Ritu 218
Ching, Dixie 196
Cho, Moon-Heum 184
Cox, Richard 134
Crespo, Pedro 232

**D**

Dale, Robert 134
De Bra, Paul 216
Desmarais, Michel 81
Dolan, Robert 172
Droschler, Stefan 95
Dy, Thomas 152

**E**

Eagle, Michael 164
Elkina, Margarita 228
Ezeife, Christie 218

**F**

Fancsali, Stephen 238
Forsyth, Carol 172
McLaren, Bruce 152
McLaughlin, Elizabeth A. 17
Fortenbacher, Albrecht 228

**G**

Galen, William 214
García-Saiz, Diego 206
Germany, Mae-Lynn 172
Geryk, Jan 103
Goldin, Ilya 73
González-Brenes, José 49
Gowda, Sujith 126
Graesser, Arthur 25, 172, 212
Griffiths, Thomas 226
Guerra, Daniel 210

**H**

Halpern, Diane 172
Harris, Thomas K. 144
Heffernan, Neil 33, 176

**J**

Jayme, Alejandra 152
Johnson, Matthew 164
Johnson, Stuart 220

**K**

Kalka, Jessica 126
Kappe, Leonard 228
Kay, Judy 234
Keshtkar, Fazel 212
Kim, Jihie 10
Kinnebrew, John 57, 65
Kocielnik, Rafal 236
Koedinger, Kenneth R. 17, 73, 196
Kortemeyer, Gerd 95
Kusbit, Gail 126

**L**

Lamar, Michelle 226
Lee, Jung In 118
Lester, James 156
Lomas, Derek 196
Lopez, Manuel Ignacio 148
Luna, J.M. 148, 180

**M**

Maldonado, Roberto Martinez 234
Manie, Nicolai 230
McCalla, Gordon 87
Shaw, Erin 10
Sidorova, Natalia 236

Mccuaig, Judi 160
Merceron, Agathe 228
Merlini, Donatella 208
Millis, Keith 172
Moldovan, Cristian 25
Molina, María De Mar 180
Morgan, Brent 212
Mostow, Jack 41, 49
Mott, Bradford 156

**N**
Naceur, Rhouma 81
Niraula, Nobal 25

**O**
Obsivac, Tomas 103
Ocumpaugh, Jaclyn 126

**P**
Pardos, Zachary 33, 169, 192
Pavlik Jr, Philip 172
Pechenizkiy, Mykola 216, 236
Peckham, Terry 87
Popelinsky, Lubos 103
Pritchard, David 95
Pursian, Andreas 228

**R**
Rafferty, Anna 226
Rau, Martina 110, 168
Rayyan, Saif 95
Ritter, Steven 196
Rivers, Kelly 144
Rodrigo, Ma. Mercedes 152
Romanczuk, Alex 134
Romero, Cristobal 148, 180
Rossi, Lisa 126
Rus, Vasile 25

**S**
Sabourin, Jennifer 156
Salvi, Aatish 126
Sarkozy, Gabor 33
Scheihing, Eliana 210
Scheines, Richard 110
Schwarzrock, Sebastian 228
Seaton, Daniel 95
Smirnov, Evgueni 230
Sprugnoli, Renzo 208
Stamper, John 17, 196
Steinhart, Jonathan 196
Sudol, Leigh Ann 144
Sun, Xiaoxun 222
Surpatean, Alexandru 230

**T**
Tan, Ling 224
Toledo, Pedro 216
Trcka, Nikola 216
Trivedi, Shubhendu 33, 192

**V**
Ventura, Cristobal
Ventura, Sebastián 148, 180

**W**
Wagner, Angela 126
Wang, Qing Yang 192
Wang, Yutao 176, 200
Warnakulasooriya, Rasil 214
Wenzlaff, Boris 228
Wixon, Michael 126

**X**
Xu, Hao 10
Xu, Yanbo 41

**Y**
Yacef, Kalina 234
Yoo, Jin Soung 184
Yudelson, Michael 188

**Z**
Zaiane, Osmar 220
Zorrilla, Marta 206