Data Mining for Individualised Hints in eLearning

Anna Katrina Dominguez, Kalina Yacef, James R. Curran

{adom0244, kalina, james}@it.usyd.edu.au
School of Information Technologies, University of Sydney, Australia.

Abstract. In this paper we present a tool where both past and current student data is used live to generate hints for students who are completing programming exercises during a national programming online tutorial and competition. These hints can be links to notes that are relevant to the problem detected and can include pre-emptive hints to prevent future mistakes. Data from the year 2008 was mined, using clustering, association rules and numerical analysis, to find common patterns affecting the learners' performance that we could use as a basis for providing hints to the 2009 students. During its live operation in 2009, student data was mined each week to update the system as it was being used. The benefits of the hinting system were evaluated through a large-scale experiment with participants of the 2009 NCSS Challenge. We found that users who were provided with hints achieved higher average marks than those who were not and stayed engaged for longer with the site.

1 Introduction

Educational Data Mining (EDM) has been applied to extract patterns from student data, but it is usually employed after the course has finished, for instance to analyse student behaviour. The idea of carrying out data mining as an integrated, functional part of the system has yet to be explored, and is outlined as future work by [1].

In this paper, we investigate whether data mining can be used to generate tailored feedback for users of an existing e-learning system without the overhead of building an ITS from scratch. We build on the NCSS Challenge[1], an annual web-based programming competition in which students are taught the basics of programming in Python. Each week they are given sets of notes and questions to answer by submitting source code. Challenge problems become very difficult in later weeks and in previous years, this had resulted in a high dropout of participants. We aim to address this issue by providing more support during the Challenge, in the form of hints and easier access to notes.

In our system, feedback is provided to users through the introduction of data mining *directly into the system loop* to provide hints. The hinting system helps students with topics that they are struggling with, by suggesting parts of the notes to reread and previous questions to revisit. The hints are generated automatically using patterns mined from both the previous year's data and the data generated by live users of the system in previous weeks. This is in contrast to the manual construction of solutions and feedback in previous systems. The hinting system is fully operational, and was used in the 2009 NCSS Challenge. We conducted an experiment to measure its effectiveness in providing

[1] http://challenge.ncss.edu.au

aid for students by presenting hints to some students and not others. We found that the system helped students considerably, resulting in a large and highly significant effect on the marks and retainment of those students who received automated hints.

2 Related work

The excellent review by Shute of formative feedback [2] provides guidelines on what to look for and what to avoid when designing feedback, and outlines that learning is best improved when the feedback is kept as simple and focused as possible. Hints and cues fulfil this objective. The Lisp Tutor [3], ELM-ART [4] and SQLT-Web [5] are all examples of successful programming tutors. Each is able to aid student problem solving by providing feedback in the form of very specific hints and error messages, even to the point of providing correct code snippets if the student is struggling. All three systems rely on example solutions and a finite set of rules that specify what code is valid for a given task, and any deviations that will result in error can be identified quickly. Encoding such sample solutions and rules requires the formulation of pre-determined feedback for students, which can be expensive in terms of time and cost, and may not cover all scenarios. These methods are fairly domain specific, and therefore cannot be easily generalised to other systems.

While the use of data mining to aid the diagnosis of students' behaviour and ability is common, relatively little work has been done in using data mining to support student problem solving. One system that aims to do this is the Logic-ITA [6], where the system takes into account past associations of student mistakes to provide on-the-fly, proactive feedback to the students. The identification of students who are in difficulty in a programming course has been the aim of some studies such as [7] and [8], but with the aim of predicting failure or dropping out of a course, not for providing hints.

3 The NCSS Challenge site

The NCSS Challenge is an annual online programming competition that has been running for Australian high school students and teachers since 2005. The Challenge runs for five weeks during which participants are provided with notes and questions on a weekly basis, and are given feedback and marks out of 10 based on their submissions to the questions. When a user submits an answer to a question, it is evaluated by an automatic marker. This marker generates basic feedback, giving high-level descriptions of tests that the submission passed or failed to the users. We have now developed an extension to the Challenge website that provides users with additional feedback when they fail a question, in the form of hints which we describe next.

3.1 Hints

We implemented two main types of hints: "pre-emptive" and "post-failure". Pre-emptive hints are available to weak students before they submit any code for a question. Post-failure hints are provided after a student's submission failed. Both types of hints lead to a hints page, automatically generated and specific to each student and question. It suggests links for (1) topics to review, (2) similar questions to attempt and (3) easier questions to

attempt. A screenshot of a post-failure hint is shown in Figure 1. Pre-emptive ones look similar, except they do not contain any associated topics to review.

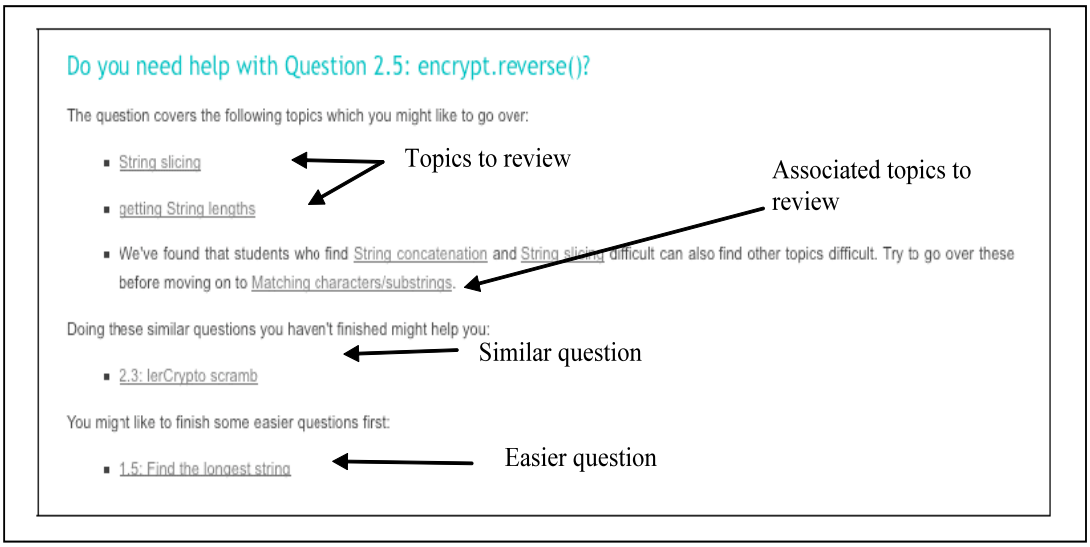

Figure 1. Screenshot of a post-failure hint page

The *topics to review* hints link to specific sections of the notes that address these topics (e.g. string slicing in Figure 1). Each section of the notes and each question were tagged with the relevant Python topics, as per a lightweight ontology that we built for that purpose, therefore allowing for the notes and questions to be related to each other. The *question* hints link to other Challenge questions selected by the data mining. The hints only suggest notes and questions that all students already have access to.

4 Data mining for the hinting system

As mentioned earlier, we used data mining as a basis for the generation and triggering of relevant hints for each student. The data came from the past 2008 Challenge and the live 2009 Challenge, including the questions, the corresponding topic tags from the ontology and the students' results and submissions. Data mining was carried out using clustering, association rule mining and simple numerical analysis, the results of which were used as various components in the final hinting system as indicated below.

Table 1. Generation methods for each part of the hint

Part of the hint	Method to generate it	Pre-emptive hints	Post-failure hints
Topics to review	Topic tags	√	√
Associated topics to review	Association rules on topics of failed submissions		√
Easier questions	Clustering questions (difficulty)	√	√
Similar questions	Clustering questions (similarity)	√	√

Table 2. Triggering mechanisms for hints

Hint	Triggered for	Method
Pre-emptive	Weak students	Clustering (students per ability)
Post-failure	All students after x failures	Simple numerical analysis

4.1 Mining the 2008 data

We mined the Challenge data from the previous year (2008) with the overall aim to (i) try to extract useful information that could be used to generate hints for our 2009 Challenge students and (ii) experiment with some techniques such as clustering to select and fine-tune our methods before using them live in 2009.

Throughout the five weeks of the 2008 Beginners Challenge, 16,814 submissions were gathered from 712 separate users. There were 25 questions in total (5 per week) that were available to the students, usually in increasing order of difficulty. Students could submit several times for the same question until successful. All these attempts were recorded, along with the mark eventually obtained by students in each question.

4.1.1 Clustering students

The aim was to group students based on their abilities. We used the K-Means clustering algorithm used in Tada-Ed [9]. For each student ID, we collated the following attributes for each of the 25 questions: whether the student attempted the question (nominal), whether the student eventually passed the question (nominal), and the marks gained for the question (numeric [0 or 5-10]). We also computed the average numbers of passed and failed questions, and the average number of submissions before the student passed a question. Clustering with these attributes produced three distinct groups, which we identified as being “strong”, “medium” and “weak” students. While this result was only relevant for the 2008 students, the effectiveness of clustering with these pre-processed attributes indicated that clustering was a viable technique for discriminating between students.

4.1.2 Clustering questions

We clustered the questions with two distinct aims: to find questions that were similar to each other and to group questions by difficulty. Our goal was to remind students of other related questions that may help them with the question they were considering at the time. We again used the K-means algorithm. The similarity-based clusters were extracted using the question metadata (topic tags). We found 5 clusters, as each of the 5 weeks of the Challenge introduced new topics. The difficulty-based clusters of questions were extracted based on the number of students who passed each question and the percentage of students who passed it that attempted it. Similarly to the clustering of students, we found three clusters, which we identified as “easy”, “medium” and “hard”. Table 3 shows the 2008 questions with performance statistics and their difficulty clusters. We found that the three groups were not grouped chronologically, but that several medium questions

appeared earlier than the last of the easy questions, and that the hard questions were interspersed through the medium ones. This meant that clustering to generate difficulty levels was worthwhile, as a simple chronological ordering would not have worked.

Table 3. The 2008 questions and their clustered difficulty rankings

Week	Question	Passed	%	Difficulty
1	hello-world	691	98	easy
1	hello-name	652	97	easy
1	string-repeater	589	96	easy
1	fuel-consumption	461	94	easy
1	eliza-simple	418	90	easy
2	aardvark	461	91	easy
2	pig-latin-basic	434	97	easy
2	simon-says	392	95	easy
2	hailstone-numbers	353	96	easy
2	animal-wordgame	315	93	medium
3	weird-units	418	98	easy
3	leap-years	345	95	easy
3	birthday-cards-for-loop	321	96	medium
3	diamonds	278	96	medium
3	eliza-interactive	226	90	medium
4	aardvark-files	227	96	medium
4	word2sms	226	98	medium
4	find-max	213	98	medium
4	primes-contact	161	79	hard
4	animal-interactive	165	90	medium
5	tranlation-latin2english	139	92	medium
5	function-swedish	118	82	hard
5	rationals-simple	101	95	medium
5	rpn-evel-simple	87	91	medium
5	eliza-templates	38	64	hard

4.1.3 Mining associations in topics

The aim was to find association rules that indicated which topics should be mastered before another question was attempted, so that the hints could suggest topics that students should review before moving onto a more complex one. We initially aimed to generate the association rules by assigning scores for each student on the 46 tags used in the 2008 Challenge. If a student passed a question, they were given one point for each tag in the question; if they did not pass a question a point was taken away for each relevant tag. Once all the scores were calculated, positive scores were labelled as "passed" and zero or negative scores were labelled as "failed". This method, however, was too coarse-grained. If a student failed a question tagged with a large number of tags but only had problems with one or two topics, they would be penalised for all the topics. We revised this to a more fine-grained method, and mined sequences of tags that students failed on. We ordered the students' results chronologically, and kept an ordered sequence of the tags for each question they made an incorrect submission to. We then used these sequences to generate association rules. Originally, we set the support and confidence to 70%. However, this excluded many advanced topics as they occurred less often in the students' sequences. This was because they were introduced in later weeks, and as such, fewer questions were tagged with them and fewer students attempted those questions. We

therefore lowered the support and confidence to 20%, and used cosine, which has been shown to be a more appropriate evaluation metric for educational data [10]. We post-processed the rules generated by the aPriori algorithm [11] to discard rules with a cosine of less than 0.65 [10] and rules with topics out of the order in which they appeared in the notes. We only retained rules that had two topics in the antecedent and one in the consequent. We finally manually extracted the rules in which the three topics involved were related to one another to remove trivial rules. We ended up keeping 83 rules, two of which are shown in Table 4. The first rule means that students who struggle with basic arithmetic in Python and comparison operators also struggle with how to loop over a set of values, and the second one means that those who struggle with converting to integers and while loops also struggle with stopping after a number of iterations.

Table 4. Examples of association rules found

mth:arth, mth:bool:comp	→ ctrl:loop:term:vals
typ:int:cast, ctrl:loop:whl	→ ctrl:loop:term:nums

4.1.4 Numerical Analysis

Aside from the more complex data mining algorithms, we subjected the 2008 data to some simple numerical analysis to find frequencies and averages for certain aspects of the data. An important measure was to have an idea of the "give-up point", that is, the number of wrong submissions a student made before he or she stopped attempting it. To do this we found, for each question, the total number of submissions made by students who never passed the question. We then averaged this over the number of students who had attempted but not passed that question. We then computed the mean of the averages already found, which was found to be 3.7. This was used in the final system as the point at which students were presented with post-failure hints; a student would only receive such hints after making their fourth incorrect submission to a question.

5 Experiment and results with 2009 Challenge

We tested our hinting system on participants of the 2009 NCSS Challenge. We ran the experiment using the live data being generated by the Challenge participants. We evaluated through a controlled experiment whether the hinting system had a positive effect on student performance, based on their marks and the ability clusters they were grouped into.

5.1 Experiment design

The Challenge ran for five weeks from mid August to late September 2009. There was an overnight period between one week's questions being closed to submissions and the next week's questions being released, which allowed us to carry out the data mining using the live system and current participant data and upload the new clusters for the next week. This also involved finalising the tags for the questions, and clustering both the questions and the students. All data mining was carried out as described in the previous section.

1303 participants registered for the 2009 NCSS Beginners Challenge. We took the first 1000 participants to enrol to be involved in our experiment, and provided them all with hints for the first week of the Challenge. At the end of the first week, we found any participants who had not yet made any submissions, and excluded them from our experiment due to inactivity, ending with a population of 584 students. At the end of week 1 where everyone received hints, we split the students into 2 equal-sized groups: a test group, which received hints, and a control group, which did not from week 2 to 5. All 584 students were clustered based on their week 1 results. We then split each cluster in half for our hinted and control groups, based on the schools the students were registered with so students from the same school were in the same group.

At the end of each week, students were clustered according to abilities (as in 4.1.1), using the cumulative student data. Question clusters were also updated weekly. We discovered each cluster could be mapped to a specific topic when we clustered the 2008 questions. Since new topics were introduced each week, we increased the number of similarity clusters over the weeks. We created two clusters in week 1 and week 2, and then increased this weekly until there were five clusters in week 5.

Unlike the 2008 questions, in which all the data on student performance per question was known and available for clustering, the data for the 2009 Challenge was being generated during the experiment. As such, we were required to estimate the difficulties for each of the new week's questions instead of deriving them solely from the participant results. At the end of each week, we analysed the results data from that week and clustered the questions to assign their difficulty levels, then estimated and assigned a difficulty to the new questions for the next week. At the end of the next week we readjusted the difficultly level based on the results generated by the participants for those questions. While the difficulty levels sometimes needed readjustment, we were generally able to estimate the question difficulties accurately at the start of the week and could predict the difficulties that were generated at the end of the week by the clustering.

5.2 Results of 2009 clustering

The techniques and attributes of the clustering of questions and students was the same as in 2008. The final set of question clusters by similarity is presented in Table 5. These were the clusters as used for Week 5 of the 2009 Challenge.

Table 5. The 2009 questions clustered by similarity

Cluster ID	Question	CLuster ID	Question
0	file-merge snakes-n-ladders	3	anagram-super biscuit-value black-ballon-calc cellular-automata debug-longest-string debug-odd-one-out download-time hello-name hello-world pirate-count-lines
1	bfs-sudoku function-list-calculator function-pirate-translate		
2	banner-printer numbers-to-words pirate-decode-repeats shopping-list-file sort-uniq substitution-cipher sudoku-checker	4	crypto-scrambler-simple crypto-substitution-undo hollow-square pirate-yarrrr

The data for the difficulty clustering of the questions is shown in Table 6, which also includes the clusters questions were assigned to at the start of week 5. The week 5 difficulty clusters (in italics) were estimated.

Table 6. The 2009 questions and their clustered difficulty rankings

Week	Question	Week 5 Difficulty	Passed	%
1	hello-world	easy	579	99
1	hello-name	easy	563	99
1	pirate-yarrrr	easy	485	95
1	download-time	easy	404	89
1	debug-longest-string	easy	383	91
2	biscuit-value	easy	442	97
2	pirate-count-lines	easy	364	97
2	crypto-scrambler-simple	medium	282	93
2	debug-odd-one-out	easy	364	97
2	crypto-substitution-undo	medium	286	88
3	black-ballon-calc	easy	343	97
3	hollow-square	medium	305	97
3	pirate-decode-repeats	medium	212	95
3	anagram-super	medium	232	97
3	sort-uniq	medium	223	98
4	numbers-to-words	medium	272	96
4	file-merge	medium	186	94
4	substitution-cipher	medium	171	93
4	banner-printer	hard	123	95
4	sudoku-checker	hard	118	87
5	shopping-list-file	*medium*	97	92
5	function-pirate-translate	*medium*	83	95
5	function-list-calculator	*medium*	75	97
5	cellular-automata	*hard*	57	95
5	bfs-sudoku	*hard*	29	74
5	snakes-n-ladders	*hard*	17	61

5.3 Evaluation

Firstly, we measured student performance based on their average overall marks. For each of the 584 students in the experiment, we calculated their average score out of 10 for the questions they made at least one submission to. We then calculated the mean of these scores over the hinted students, and the mean of the scores for the control group. The hinted group's mean score was 4.02 (sd = 2.78), while the control group had a mean score of 3.18 (sd = 2.71). This was a difference of 0.84, i.e. an increase of 26.4%, with a significance of $p < 0.0006$ using an Approximate Randomisation test [12]. We used this because the students' marks were not normally distributed, making a t-test inappropriate. This indicates that the hints substantially helped students when solving problems and lead to a significantly higher level of correct submissions being submitted.

Table 7. The number of students who submitted at least one question per week

Week	1	2	3	4	5
Hinted	292	244	203	172	75
Control	292	220	163	120	38

We also investigated whether users who received hints were active on the site for longer than those in the control group. Table 7 shows the number of hinted and control group users who submitted an answer to at least one question per week. There were consistently more users in the hinted group who made submissions, meaning the hinted group of users had an overall higher level of participation over the five weeks of the Challenge. The hints therefore had a distinctly positive effect on students' willingness to stay engaged with the course.

To get insights on students' satisfaction with the hinting system, we presented them with a survey at the end of the course. Students were asked to rate the relevance of the hints to the questions they were answering and the topics they had difficulty with by using a five-point Likert scale. 67% of students found the topics "relevant" or "somewhat relevant" and 90% of them found the questions "relevant" or "somewhat relevant". Therefore, it is clear that as far as the users were concerned, the methods for choosing topics to present were effective. In addition, 71% of students stated they would like more hints.

Overall, the survey responses showed that the students found the hints helpful. When asked to provide comments, many of the students emphatically stated that the hints had helped them with problem solving, with students giving extremely positive comments and requests for the hints to continue in future years of the Challenge. Furthermore, one student found that the hints helped her access the notes much more effectively, which was our overall aim for the system: "*I found the tips more helpful, because when we are using the notes to solve the problem we really don't know where to go and what to do or which formula to use. But after using the hint formula we know where to go and what to use for solving the problem. So I reckon that the hint boxes were a very smart way to access the notes that can help us to solve the problems.*"

6 Future work and conclusion

Our project aimed to integrate data mining into an e-learning system to generate dynamically tailored hints for users. These hints give users immediate help by directing them to parts of the notes and questions that are relevant to questions that they find difficult. We built this hinting system for the NCSS Challenge website, using association rule mining and clustering on the data produced by live users, to update the system as it was being used. We evaluated the hinting system through a large-scale experiment conducted with participants of the 2009 NCSS Challenge. In the future, we would like to compare the effectiveness of our dynamic hints with statically generated hints.

We found that users who were provided with hints achieved a 26% higher average mark than those who were not provided with hints, with statistical significance of $p < 0.0006$. Furthermore, we found qualitative evidence through positive student feedback that the hinting system had greatly helped users. These results show that the use of data mining to provide hints as part of the system loop is extremely effective, and can be used to build intelligent systems with much less of the time and cost expenses associated with traditional ITSs.

References

1 . Romero, C. and S. Ventura, Educational Data Mining: A Survey from 1995 to 2005. *Expert Systems with Applications*, 2007. **33**(1): p. 125-146.

2 . Shute, V.J., Focus on formative feedback. *Review of Educational Research*, 2008. **78**(1): p. 153-189.

3 . Reiser, B.J. and J.R. Anderson, The LISP tutor: it approaches the effectiveness of a human tutor. *Byte*, 1985. **10**(4): p. 159-175.

4 . Weber, G. and P. Brusilovsky, ELM-ART: An adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*, 2001. **12**(4): p. 351-384.

5 . Mitrovic, A., An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education (IJAIED)*, 2003. **13**(2): p. 173-197.

6 . Merceron, A. and K. Yacef, Educational Data Mining: a Case Study, in proceedings of *Artificial Intelligence in Education (AIED2005)*, C.-K. Looi, G. McCalla, B. Bredeweg, and J. Breuker, Editors. 2005, IOS Press: Amsterdam, The Netherlands. p. 467-474.

7 . Ma, Y., B. Liu, C.K. Wong, P.S. Yu, and S.M. Lee, Targeting the right students using data mining, in proceedings of *ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000: New York, NY, USA. p. 457–464.

8 . Hamalainen, W., T.H. Laine, and E. Sutinen, Data mining in personalizing distance education courses, in proceedings of *World conference on open learning and distance education*. 2004: Hong Kong. p. 1-11.

9 . Merceron, A. and K. Yacef, TADA-Ed for Educational Data Mining. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 2005. **Volume 7, Number 1**: p. http://imej.wfu.edu/articles/2005/1/03/index.asp.

10 . Merceron, A. and K. Yacef, Interestingness Measures for Association Rules in Educational Data, in proceedings of *International Conference on Educational Data Mining*. 2008: Montreal, Canada.

11 . Agrawal, R. and R. Srikant, Fast Algorithms for Mining Association Rules, in proceedings of *VLDB*. 1994: Santiago, Chile.

12 . Chinchor, N., Statistical significance of MUC-6 results, in proceedings of *Fourth Message Understanding Conference (MUC-4)*. 1992. p. 390-395.