A Framework for Capturing Distinguishing User Interaction Behaviours in Novel Interfaces

S. KARDAN, C. CONATI
University of British Columbia, Canada

As novel forms of educational software continue to be created, it is often difficult to understand a priori which ensemble of interaction behaviours is conducive to learning. In this paper, we describe a user modeling framework that relies on interaction logs to identify different types of learners, as well as their characteristic interaction behaviours and how these behaviours relate to learning. This information is then used to classify new learners, with the long term goal of providing adaptive interaction support when behaviours detrimental to learning are detected. In previous research, we described a proof-of-concept version of this user modeling approach, based on unsupervised clustering and class association rules. In this paper, we describe and evaluate an improved version, implemented in a comprehensive user-modeling framework that streamlines the application of the various phases of the modeling process.

Key Words and Phrases: Student Modeling, Clustering, Associative Rule Mining

---

1. INTRODUCTION
Advances in HCI continuously aid the creation of novel interfaces to support education and training. Because of the novelty of these interfaces, it can be difficult to judge a priori which ensemble of user interaction behaviours are conducive to learning. Our long-term goal is to devise automatic techniques to analyze logs of the interactions with a novel application and identify classes of user types, their identifying behaviours and how these behaviours relate to learning. In addition, we want to use this information to create a user model, i.e., to automatically identify the behaviours of new users, and enable the application to provide adaptive support during interaction if the behaviours are associated with suboptimal task performance.

In previous work, we described a proof-of-concept user modeling approach that uses unsupervised clustering and class association rules to identify relevant user types/behaviours from an existing dataset, and relies on these to classify new users. In this paper, we refine that proof-of-concept into a comprehensive user-modeling framework that streamlines the phases necessary to generate a user classifier from an initial dataset of raw interaction logs. In [1] the initial approach was evaluated on an environment to support learning of AI algorithms via the exploration of interactive simulations. Here, we evaluate the new user modeling framework on the same environment but on a larger dataset (65 students vs. 24), thus providing more convincing evidence on the approach effectiveness.

After discussing related work, we illustrate the general user modeling approach, including improvements from previous versions. Next, we discuss an empirical evaluation of the framework and conclude with a discussion of future work.

2. RELATED WORK
Association rules have been widely used for off-line analysis of learners' interaction patterns with educational software. e.g., to discover (*i*) error patterns that can help improve the teaching of SQL [14]; (*ii*) similarities among exercises for algebra problem solving in terms of solution difficulty [6]; (*iii*) usage patterns relevant for revising a web based educational system spanning a complete university course [7].

Authors' addresses: S. Kardan and C. Conati, Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver, BC, V6T1Z4, Canada. E-mails: skardan@cs.ubc.ca, conati@cs.ubc.ca

Most work on using association rules for on-line adaptation has been done within research on recommender systems. In [4], for instance, association rule mining is used to match the user type with appropriate products. The main difference with our work is that in [4] there is no on-line classification. Users are "labelled" based on clusters built off-line and the labels are used to guide recommendations when these users utilize the system. In contrast, we perform online classification of new users, with the goal of eventually providing real-time adaptation. Similarly, associative classification is used in [20] to classify user requirements and generate personalized item recommendation in an e-commerce application. The main difference with our work is that the approach in [20] needs labelled data, while ours can work with unlabelled datasets.

The work by Romero et al ([16]) is the most similar to the research described here, in that the authors aim to use clustering and sequential pattern mining to recognize how students navigate through a web-based learning environment, classify them and use some teacher tuned rules for recommending further navigation links accordingly. The evaluation of this work focused on analyzing the quality of the rules generated by different algorithms, but no results have yet been presented on the classification accuracy of the proposed approach.

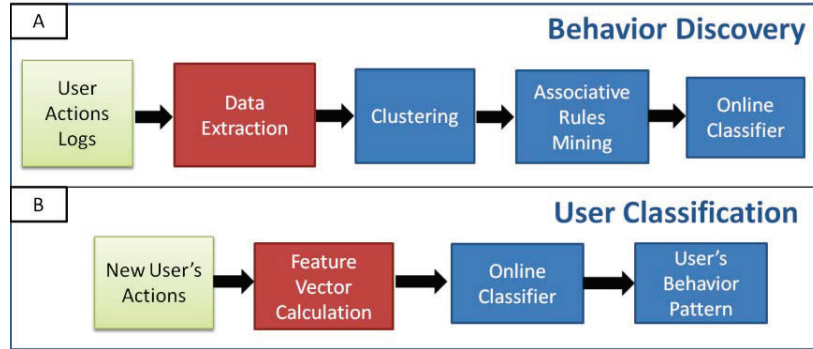## 3. GENERAL USER MODELING FRAMEWORK

Figure 1: general User Modeling Approach.

Our user modeling approach consists of major phases: *Behaviour Discovery* (Figure 1A) and *User Classification* (Figure 1B). In Behaviour Discovery, raw unlabeled data from interaction logs is preprocessed into feature vectors representing individual users in terms of their interface usage. These vectors are the input to an unsupervised clustering algorithm that groups them according to their similarity. The resulting clusters represent users who interact similarly with the interface. These clusters are then analyzed to (*i*) identify if/how they relate to learning and then (*ii*) isolate in each cluster those behaviours that are responsible for this performance. In [3] we introduced the use of Class Association Rules [18] to identify the interaction behaviour characteristics of each cluster.

Understanding the effectiveness of a user's interaction behaviours is useful in itself for revealing to developers how the application can be improved e.g. [10]. However, we also want to use these behaviours to guide automated adaptive support during interaction. Thus, the clusters and behaviours identified in the Behaviour Discovery phase are used to build an on-line classifier user model. In the User Classification phase (Figure 1B), this classifier is used to assess the performance of a new user based on her interaction behaviours. This assessment will eventually guide adaptive interventions that encourage effective interaction behaviours and prevent detrimental ones.

To test this approach, we generated a proof-of-concept version based on off-the shelf components and simplistic parameter settings. Following the encouraging results we obtained with this initial version [1], we have refined all framework components and implemented them in a Python-based unifying framework that streamlines the application of the various phases of the user modeling process. In the next few sections, we describe the most salient improvements we have made to the framework.

3.1 DATA EXTRACTION

The first step in *behaviour discovery* phase is to create a set of data-points from user interaction logs. Currently, our data-points are vectors of features consisting of statistical measures that summarize the user's actions in the interfaces (e.g. action frequencies; time interval between actions). Another approach is to create data-points from sequence mining. This approach is useful when actions order is important to identify relevant behaviours, and has been successfully applied when there are few high-level types of actions (e.g. a successful attempt on the first step of a problem, asking for hints, etc.) e.g. in [12,17]. These conditions do not apply to the test-bed educational environments we have used so far (described later), i.e. interactive simulations with many fine-grained interface actions that can be done in any order, which makes looking for recurring sequences in user actions computationally expensive without much added value.

3.2 USER CLUSTERING

In the initial version of our user-modeling approach, for clustering we used a standard implementation of the k-means algorithm [5] available in the Weka data mining package [9]. To refine the clustering step, we first experimented with other clustering algorithms available in Weka, including Hierarchical Clustering and Expectation Maximization [5]. None of these alternatives, however, substantially outperformed k-means. We thus decided to retain k-means as the clustering algorithm for our approach, but devised a method to ensure faster convergence to a good set of clusters.
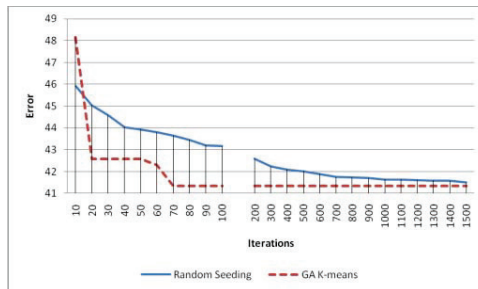

Figure 2 - Convergence of GA K-means compared to Random Seeding (K=2)

One of the issues when using the k-means is setting good initial centroids, so that the algorithm can quickly converge to a stable set of clusters with small inter-cluster error. The implementation available in Weka tended to converge slowly on the dataset we used as a test-bed for this research (described in a later section). We thus experimented with Genetic Algorithms (GA) to initialize the centroids in k-means, based on an approach suggested in [11]. This approach relies on using "chromosomes" to mold initial cluster centroids as needed. These chromosomes represent different initial values for each feature and of the initial centroids. Through mutation and crossover, in each iteration, new initial centroids are generated and the ones with lower corresponding inter-cluster error for the resultant clusters are retained for next iteration.

In our user modeling tasks, we have 21 continuous features, so the method proposed in [11] is inefficient because it requires chromosomes with too many extra bits to discretize the features without major loss of information. We thus changed the approach in [11] as follows. We generate a random population of 100 initial chromosomes, each used to generate a set of centroids that initialize a different run of k-means. We then select the half of the chromosomes that led to clusters with the lowest inter-cluster error and use

these to generate the next generation using crossover (i.e. selecting two chromosomes and choosing the upper half bits of one chromosome and the lower half of the other chromosome to form a new one) and mutation (i.e. selecting a chromosome and randomly changing one of its bits). We repeat the process until there is no improvement for a certain number of generations or we reach the maximum number of iteration limit. Our experimental results show that, although this approach does not guarantee finding the global minimum for the inter-cluster error, it converges faster than the standard random seeding method. Figure 2 for instance, compares the performances of GA k-means and the k-means from Weka on the dataset that is the test-bed for this research (averaged over 30 different runs). GA k-means converges after 100 iterations, while the standard seeding method does not reach that same error level even after 1500 iterations (here, iterations are the number of times that basic k-means is used for both cases).

3.3 ASSOCIATION RULE MINING TO DESCRIBE USER BEHAVIOURS

In our user modeling framework, association rule mining is used to identify the interaction behaviours that characterize each of the clusters found in the clustering phase. We use the Hotspot algorithm [9] to perform association rule mining on our clusters. Hotspot inspects the training data and generates the association rules corresponding to a class label (a specific cluster, in our case) in the form of a tree. For instance, two sample generic rules derived from the same tree branching could be as follows:

If *Action A frequency = High* → Cluster X	If *Action A frequency = High* **and** *Action B frequency = Low* → Cluster X

The algorithm has three parameters that influence the type and number of rules generated: the minimum level of support requested for a rule to be considered relevant (where support for rule $X \rightarrow Y$ is defined as the percentage of data points satisfying both X and Y in the dataset); the tree branching factor, influencing how many new rules can be generated from an existing one by adding a new condition; the minimum improvement in confidence needed for creating a new tree branch (where confidence for rule $X \rightarrow Y$ is the probability that Y occurs when X does). Essentially, the goal is to find a few rules that characterize as many elements in the cluster as possible and provide an easily understandable explanation of users' behaviours for each cluster.

Improvements on Rule Mining: rule generation. In the original version of the approach [1], we kept the Hotspot's default values for minimum improvement (0.01) and branching factor (2), and experimented with level of support within each cluster as a criterion to filter out rules [13]. In the new framework, we added a functionality to experiment with a variety of parameter settings. We also modified the criterion for filtering out rules so that, when there is a set of rules derived from the same tree branching, rules closer to the root and with low confidence are discarded. The rationale behind this choice is that rules with low confidence include interaction behaviours that are not representative of a specific cluster (i.e., these behaviours are observed in more than one cluster), and thus they tend to weaken the classification ability of the rule set as a whole (more detail on this point is provided in the section on user classification).

Improvements on Rule Mining: features discretization: Class association rules mining algorithms generally work with both discrete and continuous values. The attributes that describe the user interaction behaviours in our user modeling tasks are continuous, but they need to be discretized, otherwise they would produce a large number of very fine-grained rules, unsuitable for classification. Choosing the appropriate number of bins for feature discretization involves a trade-off between information loss (having too few bins)

and generating overly specific rules too detailed to capture meaningful patterns (having too many bins). While in [3] we chose a simple binary discretization, here we experimented with higher number of bins and empirically set the maximum number of bins to 7. In online user classification, as explained in the next section, the number of user actions observed is limited and it is possible that the feature values calculated for a user fall in different adjacent bins overtime, higher number of bins makes the classifier more tolerant to these fluctuations (i.e. a minor change in a feature value does not trigger a changing the label assigned to the user).

3.4 USER CLASSIFICATION

In the *user classification* phase, as new users interact with the system they are classified in real-time into one of the clusters generated by the behaviour discovery phase, based on which association rules match their behaviours. The use of association rules to construct a classifier is called Associative Classification Mining or Associative Classification [18]. Algorithms for Associative Classification usually, generate a complete set of class association rules (CARs) from training data, and then prune this initial set to obtain a subset of rules that constitute the classifier. When a new unknown object (a user in our case) is presented to the classifier, it is compared to a number of CARs and its class is predicted based on a measure that summarizes how well the user matches the CARs for each class. In the first version of our approach, the classification measure was simply the number of CARs satisfied for each cluster. This means that all rules were considered equally important for classification, failing to account for the fact that some rules with limited class support (i.e., applicable to fewer members of the class compared to others) should be considered with caution when deciding the class label of a user. In the current version, we improved the classification measure based on an approach that assigns a value to each rule, and calculates class membership scores based on the values of the satisfied rules that apply to a class (e.g. [19]). We used a variant of this approach where, instead of calculating membership scores based only on the satisfied rules, all of the CARs that represent a cluster are used. The rationale behind this choice is that, in our user modeling task the rules that do not apply to the new instance are also important for determining the final label. For instance, it is important to penalize the score of a class c when a major rule (which applies to most of the c's members) is not satisfied by a new instance, even if a less distinctive rule for c applies to it. Accordingly, the membership function we adopted returns a score S_A for a given class A as follows:

$$(1) \quad S_A = \frac{\sum_{i=1}^{m} Test(r_i) \times W_{r_i}}{\sum_{i=1}^{m} W_{r_i}}, Test(r_i) = \begin{cases} 1 & if \ r_i \ is \ satisfied \\ 0 & otherwise \end{cases}$$

Where r_i's are the m rules selected as representative for class A, W_{r_i} is the corresponding rule weight (based on a measure explained below), and $Test(r_i)$ is the function that tests the applicability of a rule to a given instance. We tried different measures from the literature to define W_{r_i} [8] (including confidence, support, conviction and leverage) and found confidence to be the measure that generates the best classification accuracy.

4. FRAMEWORK IMPLEMENTATION

We implemented the user modeling framework as a toolset of modules that automate most of the process of going from interaction logs to generating the rule-based classifier and the adaptation rules. Most modules are implemented in Python, with some external calls to Weka through a command line interface (please note that the used functionalities from Weka are standard algorithms such as association rule mining, and can be replaced by any other standard tool or implemented internally and are transparent to the final user of the framework).

First, the **preprocessing module** reads the time stamped action logs and calculates the feature vectors. Next, the GA k-means **clustering module** generates the clusters and assigns labels to each user. The **discretization module** finds the optimal number of bins and discretizes the feature vectors (this module uses Weka to run the rule-based classifiers for finding the best number of bins). The discretized dataset is passed, along with the generated clusters, to the **rule generation module** for association rule mining and rule pruning. This module uses the Hotspot algorithm from Weka and, for each cluster, looks for the optimal settings from a set of predefined values for each of the three Hotspot parameters (i.e. minimum support, confidence improvement threshold and branching factor). The last module (**classifier**) parses the generated rules and builds a classifier that gets a new feature vector and returns the computed label. We implemented a **classifier evaluation module** that uses LOOCV and all the aforementioned modules to evaluate the classifier on available datasets, as follows.

For each fold of the LOOCV, a sub-module of **classifier evaluation** feeds the test user's data into the classifier trained on the reduced dataset, by incrementally updating the feature vector representing the interaction behaviours of this user. Predictions are then made for the incoming vector as described earlier. A second sub-module computes the accuracy of the classifier by checking (after each action in the user's log) whether the test user is correctly classified into its original cluster.

5. EVALUATION

We validated the current user-modeling framework on the AIspace CSP applet, the same interactive system we used to test previous versions. However, a larger dataset was generated for testing, which is described after illustrating the CSP applet.

5.1 THE AISPACE CSP APPLET

The Constraint Satisfaction Problem (CSP) Applet is part of a collection of interactive visualizations for learning common Artificial Intelligence algorithms, called AIspace [2]. Algorithm dynamics are demonstrated on graphs by using color and highlighting, and state changes are reinforced through textual messages (see Figure 3 for an example).
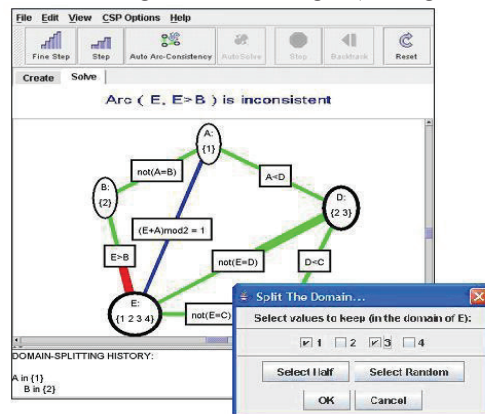

Figure 3 - CSP applet with example CSP problem

A CSP consists of a set of variables, their domains and a set of constraints on legal variable-value assignments. The goal is to find an assignment that satisfies all constraints. The CSP applet illustrates the Arc Consistency 3 (AC-3) algorithm for solving CSPs represented as networks of variable nodes and constraint arcs. AC-3 iteratively makes

individual arcs consistent by removing domain values inconsistent with a given constraint until all arcs have been considered and the network is consistent. Then, if there is still a variable with more than one value, a procedure called domain splitting is applied to that variable to split the CSP into disjoint cases so that AC-3 can recursively solve each case. The CSP applet provides mechanisms for interactive execution of the AC-3 algorithm, accessible through the toolbar shown at the top of Figure 3 or through direct manipulation of graph elements. Here we provide a brief description of these mechanisms necessary to understand the results of applying our student modeling approach to this environment:

- *Fine Stepping.* Cycles through three detailed algorithm steps: selecting an arc, testing it for consistency, and removing variable domain values when necessary.
- *Direct Arc Clicking.* Allows the user to decide which arc to test, and then performs three *Fine Steps* on that arc to make it consistent.
- Auto Arc Consistency (Auto AC). Automatically Fine Steps through the network.
- Stop. Stops Auto AC.

- *Domain Splitting (DS)*. Allows the user to select a variable domain to split, and specify a sub-network for further application of AC-3.
- *Backtracking*. Recovers the alternative sub-network set aside by *DS*.
- *Resetting*. Resets the CSP network to its initial state.

In the following sections, we describe the performance of our user-modeling framework to create a classifier user model for the CSP applet. This model will eventually be used to provide adaptive interventions for students who do not learn well while using this environment. We evaluated our framework along several dimensions.

5.2 DATA COLLECTION

Data for our evaluation was collected via user studies based on the experimental protocol for the CSP applet described in [1]. University students, who were familiar with basic graph theory but had not taken an AI course, were introduced to the AC3 algorithm and then took a pretest on the topic. Next, each participant used the applet on two CSP problems and wrote a posttest. The resulting dataset includes action logs for 65 users (compared to 24 users in [3]), totalling 13,078 actions over 62,752 seconds of interaction. From these logs, we calculated: (*i*) usage frequency of each interface action (*ii*) mean and standard deviation of latency between actions. Average latency is an indicator of the time spent reflecting after an action and planning for the next one, while standard deviation of latency tells if the user was consistent or selective in amount of pausing after each action. Since we have 7 interface actions the calculated feature vectors are 21-dimentional.

5.3 RESULTS: RELATION OF CLUSTERS TO LEARNING

First, we want to see if/how the discovered clusters relate to user learning. From the study test scores we computed the proportional learning gains for each student (in percentage), and then analyzed the clusters detected for K=2 (we used C-index as described in [15] to determine the optimal number of clusters for the data), to see if there is any significant difference with regard to learning gains. An independent samples t-test revealed a significant difference in the learning gain between the two clusters ($p = .03 < .05$), with a medium effect size (Cohen $d = .47$). We refer to these clusters as High (n = 18, M = 61.32, SD = 27.38) and Low (n = 47, M = 39.28, SD = 62.06) Learners (HL and LL). There is no significant difference between the average pretest scores of LL and HL ($p = .19$), indicating that behaviour patterns of the HL group have an impact on their learning.

5.4 RESULTS: USEFULNESS OF ASSOCIATION RULES FOR ADAPTATION

We also want to verify whether the rules generated by the framework can be used to define adaptive interventions. Table I shows a subset of the representative rules for the

HL and LL clusters in our experiment, where we report the preconditions for each rule but leave out the consequence. The table also shows, for each rule, its level of confidence (*conf*), and support within its cluster (*class cov*). These rules were generated by discretizing the feature vectors in our dataset into seven mutually exclusive ranges (bins), as explained earlier.

Direct Arc Click frequency appears in Rule1 for the HL cluster, with value in the highest bin, while it appears in Rule 3 for LL with the lowest value, indicating that LL members use *Direct Arc Click* much less than HL members. The high class coverage of Rule1 for HL (100%) indicates that high frequency of *Arc Click* pertains to all high learners, and thus it would be beneficial to trigger this behaviour for students who otherwise would not engage in it. Low values of *Direct Arc Click Pause* average and standard deviation in Rule1 and Rule2 for LL suggest that, even when they do select arcs proactively, LL students consistently spend little time thinking about this action's outcome. Finally, the high level of confidence of Rule1 for HL (100 %) indicates that, this rule will have high impact in classifying new users as per equation (1).

The above observations suggest, for instance, the following adaptation rules for the CSP applet. "`IF` user is classified as a LL and is using *Direct Arc Click* very infrequently `Then` give a hint to prompt this action"; "`IF` user is classified as a LL and pauses very briefly after a *Direct Arc Click* `Then` intervene to slow down the student"

Table I. The representative rules for HL and LL clusters

Rules for HL cluster[*]:
Rule1: Direct Arc Click frequency = Highest (Conf =100%, Class Cov = 100%)
Rule5: Domain Split frequency = Highest **and** Auto AC frequency = Lowest **and** Fine Step Pause Avg = Highest (Conf = 50%, Class Cov = 50%)
∟ **Rule8:** Domain Split frequency = Highest **and** Auto AC frequency = Lowest **and** Fine Step Pause Avg = Highest **and** Reset frequency = Lowest (Conf = 65%, Class Cov = 76.47%)

Rules for LL cluster[*]:
Rule1: Direct Arc Click Pause Avg = Lowest (Conf =100%, Class Cov = 100%)
Rule2: Direct Arc Click Pause STD =Lowest (Conf = 95.83%, Class Cov = 95.8%)
Rule3: Direct Arc Click frequency = Lowest (Conf = 93.48%, Class Cov=93.5%)
∟ **Rule4:** Direct Arc Click frequency = Lowest **and** Direct Arc Click Pause Avg = Lowest (Conf =100%, Class Cov=100%)

[*] Conf= Confidence; Avg= Average; Class Cov= Class Coverage, STD = Standard Deviation

As another example, consider Rule8 for the HL cluster. This rule indicates that HL members (*i*) use *Auto AC* action sparsely. Recall that this action quickly runs AC3 to completion, and thus is not very useful for learning how the algorithm works; (*ii*) perform *Domain Split* (the most advanced step in the algorithm) frequently (*iii*) spend the highest average time thinking about each *Fine Step* they take. This is an ensemble of effective behaviours that should be encouraged in an adaptive version of the CSP applet. Taken individually, these behaviours don't show a statistically significant difference between LL and HL and thus would not be identified as relevant by a pair wise analysis of features (as performed, for instance, in [1]).

In summary, this section illustrates that the rules generated by our framework are informative and can be used for generating real-time adaptive interventions. These interventions, however, are appropriate only if the classifier user model can recognize which users need them. Thus, the next section discusses classifier's performance.

5.5 RESULTS: PERFORMANCE ON USER CLASSIFICATION
We used LOOCV, as explained earlier, to evaluate the accuracy of our rule-based

classifier and compare it against: (i) a baseline that always predicts the most likely label (LL in our dataset); (ii) the best achieving classifier among various complex classifiers available in Weka, i.e., the Random Subspace meta-classifier using C4.5 as the base classifier; (iii) the classifier obtained with the earlier version of the framework [1] (old rule-based classifier in Figure 4). Note that all these four classifiers use the categories learned via the unsupervised process described in sections 3.1 though 3.4. We also want to compare our approach against a fully supervised approach that starts from categories defined based on the available learning gains. For this, we calculated the median of the learning gains and labelled the students above the median as high learners and others as low learners. We then trained and tested a C4.5 classifier with these new labels.

Figure 4 shows the overtime average accuracy of these five classifiers, both in terms of percentage of correct classifications for the individual clusters (LL and HL), and overall. The new rule-based classifier has the highest overall accuracy, and the differences with the other classifiers are statistically significant ($p < .001$), with a large effect size ($d > 3$). For each cluster, the accuracy of new classifier is comparable with the best competitor, but no other classifier achieves the same accuracy in both clusters.
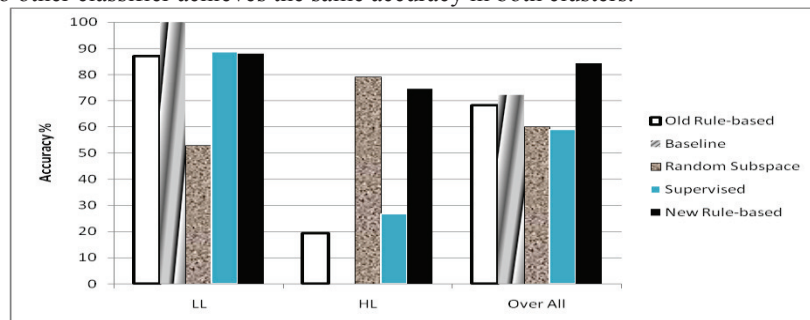

Figure 4 - The overtime average accuracy of different classifiers compared to the new rule-based classifier

Figure 5, shows accuracy of the new classifier as a function of the percentage of observed actions, both overall and for the individual clusters.
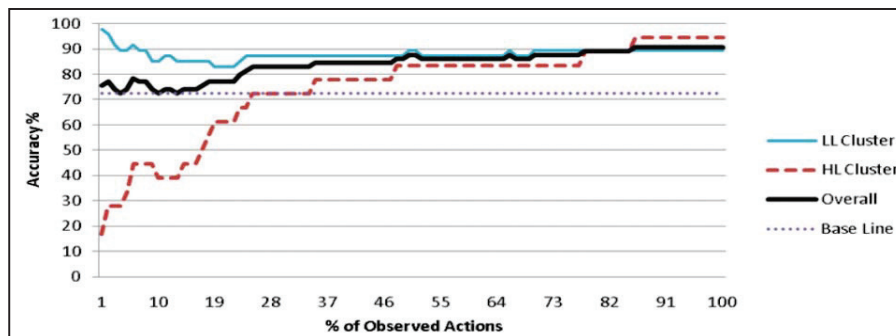

Figure 5 – Accuracy of the new rule-based classifier as a function of the percentage of observed actions

For comparison, we include the overall accuracy of the baseline, which is the best performing classifier after ours. The new rule based classifier reaches a relatively high accuracy in early stages of the interaction which is very important when the goal is to provide adaptive interventions to improve the user experience with the educational software. The overall accuracy of the new classifier becomes consistently higher than all the other classifiers before observing 20% of user actions, and accuracy on each cluster goes above 80% after seeing about 50% of the actions, while the baseline consistently misclassifies high learners throughout.

6. CONCLUSION AND FUTURE WORK

In this paper, we describe a user modeling framework that uses unsupervised clustering and Class Associating Mining to discover and recognizes relevant interaction patterns during student interaction with educational software. The framework improves a previous proof-of-concept approach by adding functionalities for more efficient clustering and more principled selection of some of the required parameters. An empirical evaluation of the framework provides evidence that it can both cluster users into meaningful groups, as well as classifying new users accurately. More importantly, the framework generates rules that provide a fine grained description of common behaviours for users in different clusters. These rules appear to be suitable to guide adaptive interventions targeted at improving interaction effectiveness. The next step of this work will be to add these adaptive interventions to the educational software we have been using as a test-bed for this research, an interactive simulation to help students understand an algorithm for constraint satisfaction. We also plan to use the framework for generating classifier user models for other educational software developed in our lab, including interactive simulations for other AI algorithms and an educational game for mathematical skills.

REFERENCES

1. AMERSHI, S. AND CONATI, C. 2009. Combining Unsupervised and Supervised Machine Learning to Build User Models for Exploratory Learning Environments. J. of Educational Data Mining 1, 1, 2.
2. AMERSHI, S., CARENINI, G., CONATI, C., MACKWORTH, A. AND POOLE, D. 2008. Pedagogy and Usability in Interactive Algorithm Visualizations - Designing and Evaluating CIspace. Interact Comput. 20, 1, 64-96.
3. BERNARDINI, A., AND CONATI, C. 2010. Discovering and Recognizing Student Interaction Patterns in Exploratory Learning Environments. Proc. ITS2010, Springer, 125-134.
4. CHANGCHIEN, S.W., AND LU, T. 2001. Mining association rules procedure to support on-line recommendation by customers and products fragmentation. Expert Systems with Applications 20, 325-335
5. DUDA, R. O., HART, P. E., AND STORK, D., G. 2001. Pattern Classification. New York: Wiley- Interscience.
6. FREYBERGER, J., HEFFERNAN, N., AND RUIZ, C. 2004. Using association rules to guide a search for best fitting transfer models of student learning. Workshop on analyzing student-tutor interactions logs to improve educational outcomes at ITS conference, 1-4.
7. GARCÍA, E., ROMERO, C., VENTURA, S., AND CASTRO, C. D. 2009. An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. UMUAI 19, 99-132.
8. GENG, L., AND HAMILTON, H. J. 2006. Interestingness measures for data mining: A survey. ACM Comput. Surv. 38, 3, 9.
9. HALL, M., EIBE, F., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., WITTEN, I.H. 2009. The WEKA data mining software: an update. SIGKDD Explor. **11**(1), 10–18.
10. HUNT, E., AND MADHYASTHA, T. 2005. Data Mining Patterns of Thought. In Proc. the AAAI Workshop on Educational Data Mining.
11. KIM, K. AND AHN, H. 2008. A recommender system using GA k-means clustering in an online shopping market. Expert Syst. Appl. 34, 2, 1200-1209.
12. KÖCK, M., PARAMYTHIS, A. 2011. Activity sequence modelling and dynamic clustering for personalized e-learning. User Model User-Adap Inter 21:51–97.
13. LIU, B., HSU, W., AND MA, Y. 1999. Mining association rules with multiple minimum supports. In Proc. KDD '99. ACM, 337-341
14. MERCERON A., AND YACEF K. 2003. A web-based tutoring tool with mining facilities to Improve Teaching and Learning, AIED 2003, 201-208.
15. MILLIGAN G. W., AND COOPER, M. C. 1985. An examination of procedures for determining the number of clusters in a data set, Psychometrika , 50, 2, 159-179.
16. ROMERO, C., VENTURA, S., ZAFRA, A., AND DE BRA, P. 2009. Applying Web usage mining for personalizing hyperlinks in Web-based adaptive educational systems. Comput. Educ. 53, 3, 828-840.
17. SHANABROOK, D. H., COOPER, D. G., WOOLF, B. P., ARROYO, I. 2010. Identifying High-Level Student Behaviour Using Sequence-based Motif Discovery. In Proceedings of EDM'2010.191-200.
18. THABTAH, F. 2007. A review of associative classification mining. Knowl. Eng. Rev. 22, 1, 37-65.
19. YIN, X., AND HAN, J. 2003. CPAR: Classification based on predictive association rules. In Proc. of the Third SIAM International Conference on Data Mining, SIAM, 208-217.
20. ZHANGA, Y., AND JIAO, J. 2007. An associative classification-based recommendation system for personalization in B2C e-commerce applications, Expert Syst. Appl. 33, 2, 357-367.