Ryan Shaun Joazeiro de Baker
Tiffany Barnes
Joseph E. Beck (Eds.)

Educational Data Mining 2008

The 1st International Conference on Educational Data Mining
Montréal, Québec, Canada, June 20-21, 2008
Proceedings

Preface

Educational Data Mining (EDM) is the process of converting raw data from educational systems to useful information that can be used by educational software developers, students, teachers, parents, and other educational researchers. EDM is also an emerging discipline, concerned with developing methods for exploring the unique types of data that come from educational settings, and with using those methods to better understand students and the settings in which they learn. Whether educational data are taken from students' use of interactive learning environments, computer-supported collaborative learning, or administrative data from schools and universities, it often has multiple levels of meaningful hierarchy, which often need to be determined by properties in the data itself, rather than in advance. Issues of time, sequence, and context also play important roles in the study of educational data.

The First International Conference on Educational Data Mining (EDM2008) brings together researchers from computer science, education, psychology, psychometrics, and statistics to analyze large data sets to answer educational research questions. The increase in instrumented educational software, as well as state databases of student test scores, has created large repositories of data reflecting how students learn. The EDM conference focuses on computational approaches for using those data to address important educational questions. The broad collection of research disciplines ensures cross-fertilization of ideas, with the central questions of educational research serving as a unifying focus. This annual conference emerges from preceding EDM workshops at the AAAI, AIED, EC-TEL, ICALT, ITS, and UM conferences.

This has been an exciting year for Educational Data Mining, with the first international conference for this emerging discipline, and the launch of the Journal of Educational Data Mining (JEDM), with Kalina Yacef as journal editor. We look forward, as well, to EDM2009, which will occur in Cordoba, Spain.

45 papers were submitted to EDM2008. After much hard work by our program committee and reviewers, 17 submissions were accepted as full papers (37.8% acceptance rate). 14 other submissions were accepted as Young Researcher Track (YRT) papers or poster papers, and appear in the proceedings in shortened form. These papers cover a rich range of topics, including students' interactions with learning software, meta-cognition, skill modeling, adaptive testing, data repositories, metrics for comparing association rules. All of these papers will appear both on the web, at www.educationaldatamining.org, as well as in the physical proceedings.

We would like to thank the University of Quebec at Montreal (UQAM) and the Machine Learning Department at Carnegie Mellon University for their generous sponsorship of EDM2008. We would like to thank the members of the program committee, steering committee, the reviewers, and the invited speakers for their enthusiastic help in putting this conference together. We would also like to enthusiastically thank Michel Desmarais, Arnon Hershkovitz, and Roger Nkambou for their essential role in making this conference happen. Local arrangements and webmastering are often thankless tasks, and their work was vital. And of course, we would like to thank Tiffany Barnes, the conference chair, for her leadership in organizing the conference.

Ryan S.J.d. Baker and Joseph E. Beck
Program Chairs

Organization

Conference Organization

Conference Chair:	Tiffany Barnes, University of North Carolina at Charlotte, USA
Program Chairs:	Ryan S. J. d. Baker, Carnegie Mellon University, USA
	Joseph E. Beck, Worcester Polytechnic Institute, USA
Local Arrangements Chairs:	Michel Desmarais, Ecole Polytechnique de Montreal, Canada
	Roger Nkambou, Université du Québec à Montréal (UQAM), Canada
Web Chair:	Arnon Hershkovitz, Tel Aviv University, Israel

Program Committee

Anjo Anjewierden, University of Twente, The Netherlands
Esma Aïmeur, University of Montreal, Canada
Ivon Arroyo, University of Massachusetts Amherst, USA
Bettina Berendt, University of Berlin, Germany
Christophe Choquet, Université du Maine, France
Cristina Conati, University of British Columbia, Canada
Janice Gobert, Worcester Polytechnic Institute, USA
Neil Heffernan, Worcester Polytechnic Institute, USA
Brian Junker, Carnegie Mellon University, USA
Judy Kay, University of Sydney, Australia
Kenneth Koedinger, Carnegie Mellon University, USA
Brent Martin, Canterbury University, New Zealand
Noboru Matsuda, Carnegie Mellon University, USA
Gord McCalla, University of Saskatchewan, Canada
Bruce McLaren, Deutsches Forschungszentrum für Künstliche Intelligenz, Germany
Tanja Mitrovic, Canterbury University, New Zealand
Cristóbal Romero Morales, Cordoba University, Spain
Mykola Pechenizkiy, Eindhoven University of Technology, Netherlands
Kaska Porayska-Pomsta, London Knowledge Laboratory, UK
Carolyn Rosé, Carnegie Mellon University, USA
Carolina Ruiz, Worcester Polytechnic Institute, USA
Sebastián Ventura Soto, Cordoba University, Spain
Steven Tanimoto, University of Washington, USA
Silvia Viola, Universita' Politecnica delle Marche, Italy
Kalina Yacef, University of Sydney, Australia
Osmar Zaiane, University of Alberta, Canada

Steering Committee

Esma Aïmeur, University of Montreal, Canada
Ryan S.J.d. Baker, Carnegie Mellon University, USA
Tiffany Barnes, University of North Carolina at Charlotte, USA
Joseph E. Beck, Worcester Polytechnic Institute, USA

Neil Heffernan, Worcester Polytechnic Institute, USA
Brian Junker, Carnegie Mellon University, USA
Silvia Viola, Università Politecnica delle Marche, Italy
Kalina Yacef, University of Sydney, Australia

Additional Reviewers

Hao Cen, Carnegie Mellon University, USA
Albert Corbett, Carnegie Mellon University, USA
Rebecca Crowley, University of Pittsburgh Medical Center, USA
Mingyu Feng, Worcester Polytechnic Institute, USA
Sabine Graf, National Central University, Taiwan
Michael Heilman, Carnegie Mellon University, USA
Cecily Heiner, University of Utah, USA
Jean-Mathias Heraud, University of Savoie, France
Arnon Hershkovitz, Tel Aviv University, Israel
Benoit Lemaire, University of Grenoble, France
Frank Linton, MITRE, USA
Manolis Mavrikis, London Knowledge Laboratory, UK
Agathe Merceron, University of Berlin, Germany
Jack Mostow, Carnegie Mellon University, USA
Philip Pavlik, Carnegie Mellon University, USA
Genaro Rebolledo-Mendez, Coventry University, UK
Mercedes Rodrigo, Ateneo de Manila University, Philippines
Amy Soller, Institute for Defense Analyses, USA
John Stamper, University of North Carolina at Charlotte, USA
Ronald Stevens, University of California, Los Angeles, USA
Titus Winters, DirecTV, USA

Table of Contents

Full papers

Posters

Young Researchers' Track (YRT) Posters

Data Mining Algorithms to Classify Students

Cristóbal Romero, Sebastián Ventura, Pedro G. Espejo and César Hervás
{cromero, sventura, pgonzalez, chervas}@uco.es
Computer Science Department, Córdoba University, Spain

Abstract. In this paper we compare different data mining methods and techniques for classifying students based on their Moodle usage data and the final marks obtained in their respective courses. We have developed a specific mining tool for making the configuration and execution of data mining techniques easier for instructors. We have used real data from seven Moodle courses with Cordoba University students. We have also applied discretization and rebalance preprocessing techniques on the original numerical data in order to verify if better classifier models are obtained. Finally, we claim that a classifier model appropriate for educational use has to be both accurate and comprehensible for instructors in order to be of use for decision making.

1 Introduction

The ability to predict/classify a student's performance is very important in web-based educational environments. A very promising arena to attain this objective is the use of Data Mining (DM) [26]. In fact, one of the most useful DM tasks in e-learning is classification. There are different educational objectives for using classification, such as: to discover potential student groups with similar characteristics and reactions to a particular pedagogical strategy [6], to detect students' misuse or game-playing [2], to group students who are hint-driven or failure-driven and find common misconceptions that students possess [34], to identify learners with low motivation and find remedial actions to lower drop-out rates [9], to predict/classify students when using intelligent tutoring systems [16], etc. And there are different types of classification methods and artificial intelligent algorithms that have been applied to predict student outcome, marks or scores. Some examples are: predicting students' grades (to classify in five classes: A, B, C, D and E or F) from test scores using neural networks [14]; predicting student academic success (classes that are successful or not) using discriminant function analysis [19]; classifying students using genetic algorithms to predict their final grade [21]; predicting a student's academic success (to classify as low, medium and high risk classes) using different data mining methods [30]; predicting a student's marks (pass and fail classes) using regression techniques in Hellenic Open University data [18] or using neural network models from Moodle logs [11].

In this paper we are going to compare different data mining techniques for classifying students based on both students' usage data in a web-based course and the final marks obtained in the course. We have also developed a specific Moodle data mining tool for making this task easier for instructors. The paper is arranged in the following way: Section 2 describes the background of the main classification methods and algorithms; section 3 describes the Moodle data mining tool; section 4 details the comparison of the classification techniques; finally, the conclusions and further research are outlined.

2 Background

Classification is one of the most frequently studied problems by DM and machine learning (ML) researchers. It consists of predicting the value of a (categorical) attribute (the class) based on the values of other attributes (the predicting attributes). There are different classification methods, such as:

- Statistical classification is a procedure in which individual items are placed into groups based on the quantitative information of characteristics inherent in the items (referred to as variables, characters, etc.) and based on a training set of previously labelled items [23]. Some examples of statistical algorithms are linear discriminant analysis [21], least mean square quadratic [27], kernel [21] and k nearest neighbors [21].

- A decision tree is a set of conditions organized in a hierarchical structure [25]. It is a predictive model in which an instance is classified by following the path of satisfied conditions from the root of the tree until reaching a leaf, which will correspond to a class label. A decision tree can easily be converted to a set of classification rules. Some of the most well-known decision tree algorithms are C4.5 [25] and CART [4].

- Rule Induction is an area of machine learning in which IF-THEN production rules are extracted from a set of observations [11]. The algorithms included in this paradigm can be considered as a heuristic state-space search. In rule induction, a state corresponds to a candidate rule and operators correspond to generalization and specialization operations that transform one candidate rule into another. Examples of rule induction algorithms are CN2 [8], AprioriC [17], XCS [32], Supervised Inductive Algorithm (SIA) [31], a genetic algorithm using real-valued genes (Corcoran) [10] and a Grammar-based genetic programming algorithm (GGP) [15].

- Fuzzy rule induction applies fuzzy logic in order to interpret the underlying data linguistically [35]. To describe a fuzzy system completely, a rule base (structure) and fuzzy partitions have to be determined (parameters) for all variables. Some fuzzy rule learning methods are LogitBoost [23], MaxLogitBoost [29], AdaBoost [12], Grammar-based genetic Programming (GP) [28], a hybrid Grammar-based genetic Programming/genetic Algorithm method (GAP) [28], a hybrid Simulated Annealing/genetic Programming algorithm (SAP) [28] and an adaptation of the Wang-Mendel algorithm (Chi) [7].

- Neural Networks can also be used for rule induction. A neural network, also known as a parallel distributed processing network, is a computing paradigm that is loosely modeled after cortical structures in the brain. It consists of interconnected processing elements called nodes or neurons that work together to produce an output function. Examples of neural network algorithms are multilayer perceptron (with conjugate gradient-based training) [22], a radial basis function neural network (RBFN) [5], incremental RBFN [24], decremental RBFN [5], a hybrid Genetic Algorithm Neural Network (GANN) [33] and Neural Network Evolutionary Programming (NNEP) [20].

3 Moodle Data Mining Tool

We have developed a specific Moodle data mining tool oriented for use by on-line instructors. It has a simple interface (see Figure 1) to facilitate the execution of data mining techniques. We have integrated this tool into the Moodle environment itself. In this way, instructors can both create/maintain courses and carry out all data mining processing with the same interface. Likewise, they can directly apply feedback and results obtained by data mining back into Moodle courses. We have implemented this tool in Java using the KEEL framework [1] which is an open source framework for building data mining models including classification (all the previously described algorithms in Section 2), regression, clustering, pattern mining, and so on.

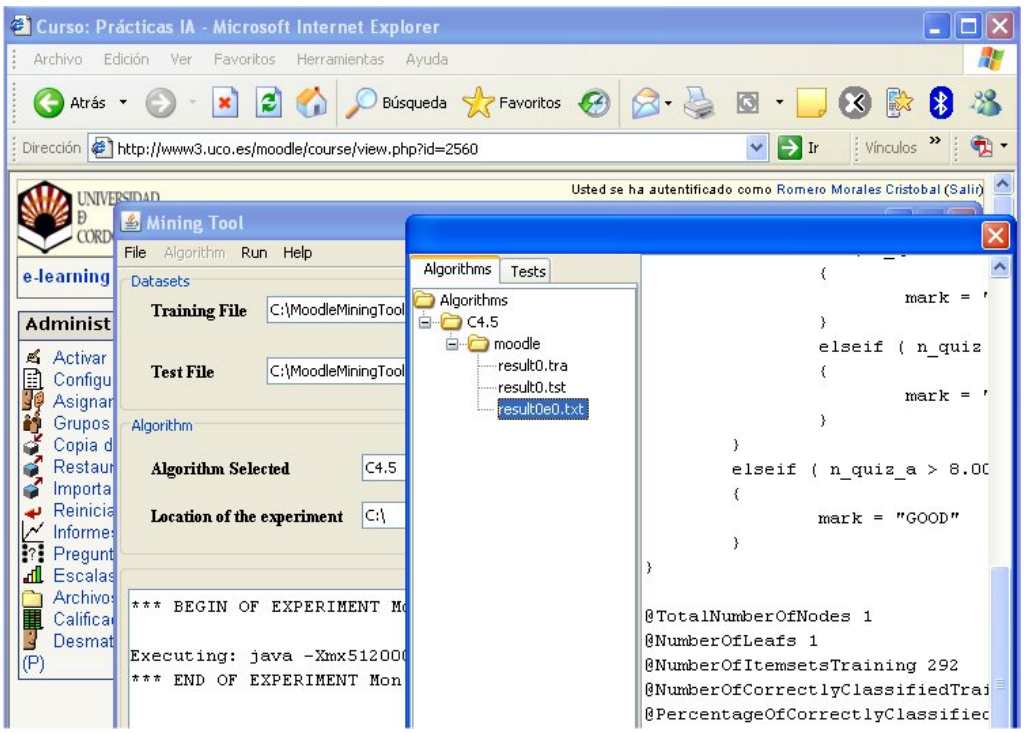


Figure 1. Moodle Data Mining Tool executing C4.5 algorithm.

In order to use it, first of all the instructors have to create training and test data files starting from the Moodle database. They can select one or several courses and one Moodle table (mdl_log, mdl_chat, mdl_forum, mdl_quiz, etc.) or create a summary table (see Table 1). Then, data files will be automatically preprocessed and created. Next, they only have to select one of the available mining algorithms and the location of the output directory. For example, in Figure 1, we show the execution of the C4.5 algorithm over a summary file and the decision tree obtained. We can see that the results files (.tra and .test files with partial results and .txt file with the obtained model) appear in a new window (see Figure 1 down in the right hand corner). Finally, instructors can use this model for decision making concerning the suitability of the Moodle activities in each specific course and also to classify new students depending on the course usage data.

4 Experimental Results

We have carried out some experiments in order to evaluate the performance and usefulness of different classification algorithms for predicting students' final marks based on information in the students' usage data in an e-learning system. Our objective is to classify students with equal final marks into different groups depending on the activities carried out in a web-based course. We have chosen the data of 438 Cordoba University students in 7 Moodle courses (security and hygienee in the work, projects, engineering firm, programming for enginnering, computer science basis, applied computer science, and scientific programming). Moodle (http://moodle.org) is one of the most frequently used free Learning Content Management Systems (LCMS). Moodle keeps detailed logs of all activities that students perform in a data base. Information is available about the use of Moodle activities and resources (assignments, forums and quizzes). We have preprocessed the data in order to transform them into a suitable format to be used by our Moodle data mining tool. First, we have created a new summary table (see Table 1) which integrates the most important information for our objective (Moodle activities and the final marks obtained in the course). Using our Moodle mining tool a particular teacher could select these or other attributes for different courses during the data preprocessing phase. The Table 1 summarises row by row all the activities done by each student in the course (input variables) and the final mark obtained in this course (class).

Table 1. Attributes used by each student.

Name	Description
course	Identification number of the course.
n_assigment	Number of assignments done.
n_quiz	Number of quizzes taken.
n_quiz_a	Number of quizzes passed.
n_quiz_s	Number of quizzes failed.
n_posts	Number of messages sent to the forum.
n_read	Number or messages read on the forum.
total_time_assignment	Total time used on assignments.
total_time_quiz	Total time used on quizzes.
total_time_forum	Total time used on forum.
mark	Final mark the student obtained in the course.

Secondly, we have discretized all the numerical values of the summary table into a new summarization table. Discretization divides the numerical data into categorical classes that are easier for the teacher to understand. It consists of transforming continuous attributes into discrete attributes that can be treated as categorical attributes. Discretization is also a requirement for some algorithms. We have applied the manual method (in which you have to specify the cut-off points) to the mark attribute. We have used four intervals and labels (FAIL: if value is <5; PASS: if value is >=5 and <7; GOOD: if value is >=7 and <9; and EXCELLENT: if value is >=9). In addition, we have

applied the equal-width method [13] to all the other attributes with three intervals and labels (LOW, MEDIUM and HIGH). Then, we have exported both versions of the summary table (with numerical and categorical values) to text files with KEEL format [1]. Next, we have made partitions of whole files (numerical and categorical files) into pairs of training and test files. Each algorithm is evaluated using stratified 10-fold cross-validation. The dataset is randomly divided into 10 disjointed subsets of equal size in a stratified way (maintaining the original class distribution). In each repetition, one of the 10 subsets is used as the test set and the other 9 subsets are combined to form the training set. In this work we also take into consideration the problem of learning from imbalanced data. We say data is imbalanced when some classes differ significantly from others with respect to the number of instances available. The problem with imbalanced data arises because learning algorithms tend to overlook less frequent classes (minority classes), paying attention just to the most frequent ones (majority classes). As a result, the classifier obtained will not be able to correctly classify data instances corresponding to poorly represented classes. Our data presents a clear imbalance since its distribution is: EXCELLENT 3.89%, GOOD 14.15%, PASS 22.15%, FAIL 59.81%. One of the most frequent methods used to learn from imbalanced data consists of resampling the data, either by over-sampling the minority classes or under-sampling the majority ones, until every class is equally represented [3]. When we deal with balanced data, the quality of the induced classifier is usually measured in terms of classification accuracy, defined as the fraction of correctly classified examples. But accuracy is known to be unsuitable to measure classification performance with imbalanced data. An evaluation measure well suited to imbalanced data is the geometric mean of accuracies per class (g-mean), defined as $g-mean=\sqrt[n]{\prod_{i=1}^{n}\frac{hits_i}{instances_i}}$, where n is the number of classes, $hits_i$ is the number of instances of class i correctly classified and $instances_i$ is the number of instances of class i. In our work, we have used random over-sampling, a technique consisting of copying randomly chosen instances of minority classes in the dataset until all classes have the same number of instances, and we use the geometric mean to measure the quality of the induced classifiers.

Finally, we have used three sets of 10-fold data files: the original numerical data, the categorical data and the numerical rebalanced data. We have carried out one execution with all the determinist algorithms and 5 executions with the nondeterministic algorithms. In Table 2 we show the global percentage of the accuracy rate and geometric means (the averages of 5 executions for nondeterministic algorithms). We have used the same default parameters for algorithms of the same type (For example, 1000 iterations in evolutionary algorithms and 4 labels in fuzzy algorithms). We have used these 25 specific classification algorithms due to they are implemented in Keel software, but there are some other classification techniquess such as bayesina networks, logistic regression, etc.

The global percentage of those correctly classified (global PCC) shows the accuracy of the classifiers (see Table 2). More than half of the algorithms obtain their highest values using original numerical data, and the other algorithms obtain them using the categorical data. This can be due to the nature and implementation of each algorithm which might be more appropriate for using numerical or categorical data. As we have seen above, it is

easier to obtain a high accuracy rate when data are imbalanced, but when all the classes have the same number of instances it becomes more difficult to achieve a good classification rate. The best algorithms (with more than 65% global PCC) with original data (numerical) are CART, GAP, GGP and NNEP. The best algorithms (with over 65% global PCC) using categorical data are the two decision tree algorithms: CART and C4.5. The best algorithms (with over 60% global PCC) with balanced data are Corcoran, XCS, AprioriC and MaxLogicBoost. It is also important to note that no algorithm exceeds 70% global percentage of correctly classified results. One possible reason for this is due to the fact that we have used incomplete data, that is, we have used the data of all the students examined although some students who did not do all the course activities did do the final exam. In particular, about 30% of our students have not used the forum or have not done some quizzes. But we have not eliminated these students from the dataset because it shows a real problem about the students' usage level of e-learning systems. So, we have used all the data although we know that this fact can affect the accuracy of the classification algorithms.

Table 2. Classification results (Global percentage of correctly classified / Geometric Mean).

Method	Algorithm	Numerical data	Categorical data	Rebalanced data
Statistical Classifier	ADLinear	59.82 / 0.00	61.66 / 0.00	59.82 / 0.00
Statistical Classifier	PolQuadraticLMS	64.30 / 15.92	63.94 / 18.23	54.33 / 26.23
Statistical Classifier	Kernel	54.79 / 0.00	56.44 / 0.00	54.34 / 0.00
Statistical Classifier	KNN	59.38 / 10.15	59.82 / 7.72	54.34 / 10.21
Decision Tree	C45	64.61 / 41.42	65.29 / 18.10	53.39 / 9.37
Decision Tree	CART	67.02 / 39,25	66.86 / 24,54	47.51 / 34,65
Rule Induction	AprioriC	60.04 / 0.00	59.82 / 0.00	61.64 / 0.00
Rule Induction	CN2	64.17 / 0.00	63.47 / 3.52	50.24 / 15.16
Rule Induction	Corcoran	62.55 / 0.00	64.17 / 0.00	61.42 / 0.00
Rule Induction	XCS	62.80 / 0.00	62.57 / 0.00	60.04 / 23.23
Rule Induction	GGP	65.51 / 1.35	64.97 / 1.16	52.91 / 12.63
Rule Induction	SIA	57.98 / 0.00	60.53 / 0.00	56.61 / 15.41
Fuzzy Rule Learning	MaxLogitBoost	64.85 / 0.00	61.65 / 0.00	62.11 / 8.83
Fuzzy Rule Learning	SAP	63.46 / 0.00	64.40 / 0.00	47.23 / 3.20
Fuzzy Rule Learning	AdaBoost	62.33 / 0.00	60.04 / 0.00	50.47 / 0.00
Fuzzy Rule Learning	LogitBoost	61.17 / 13.05	63.27 / 4.64	55.70 / 13.95
Fuzzy Rule Learning	GAP	65.99 / 0.00	63.02 / 0.00	52.95 / 26.65
Fuzzy Rule Learning	GP	63.69 / 0.00	63.03 / 0.00	53.19 / 11.97
Fuzzy Rule Learning	Chi	57.78 / 10.26	60.24 / 0.00	41.11 / 14.32
Neural Networks	NNEP	65.95 / 0.00	63.49 / 0.00	54.55 / 12.70
Neural Networks	RBFN	55.96 / 3.23	54.60 / 0.00	37.16 / 4.00
Neural Networks	RBFN Incremental	53.65 / 9.87	58.00 / 14.54	30.31 / 18.32
Neural Networks	RBFN Decremental	50.16 / 3.95	53.44 / 5.61	35.32 / 8.41
Neural Networks	GANN	60.28 / 0.00	61.90 / 4.82	53.43 / 17.33
Neural Networks	MLPerceptron	63.91 / 9.65	61.88 / 4.59	53.21 / 17.16

The geometric mean tells us about the effect of rebalancing on the performance of the classifiers obtained, since the geometric mean offers us a better view of the classification performance in each of the classes. We can see in Table 2 that the behavior depends to a great extent on the learning algorithm used. There are some algorithms which are not affected by rebalancing (Kernel, KNN, AprioriC, Corcoran, AdaBoost and LogitBoost): the two decision tree methods (CART and C4.5) give worse results with rebalanced data (C4.5) but most of the algorithms (all the rest, 17 out of 25) obtain better results with the rebalanced data. Thus we can see that the rebalancing of the data is generally beneficial for most of the algorithms. We can also see that many algorithms obtain a value of 0 in the geometric mean. This is because some algorithms do not classify any of the students correctly into a specific group. It is interesting to see that it only happens to the group of EXCELLENT students (EXCELLENT students are incorrectly classified as GOOD and PASS students). But in education this is not very dramatic after all since the most important thing is to be able to distinguish perfectly between FAIL students and PASS students (PASS, GOOD and EXCELENT).

On the other hand, in our educational problem it is also very important for the classification model obtained to be user friendly, so that teachers can make decisions about some students and the on-line course to improve the students' learning. In general, models obtained using categorical data are more comprehensible than when using numerical data because categorical values are easier for a teacher to interpret than precise magnitudes and ranges. Nonetheless, some models are more interpretable than others:

- Decision trees are considered easily understood models because a reasoning process can be given for each conclusion. However, if the tree obtained is very large (a lot of nodes and leaves) then they are less comprehensible. A decision tree can be directly transformed into a set of IF-THEN rules that are one of the most popular forms of knowledge representation, due to their simplicity and comprehensibility. So, C4.5 and CART algorithms are simple for instructors to understand and interpret.

- Rule induction algorithms are normally also considered to produce comprehensible models because they discover a set of IF-THEN classification rules that are a high-level knowledge representation and can be used directly for decision making. And some algorithms such as GGP have a higher expressive power allowing the user to determine the specific format of the rules (number of conditions, operators, etc.).

- Fuzzy rule algorithms obtain IF-THEN rules that use linguistic terms that make them more comprehensible/interpretable by humans. So, this type of rules is very intuitive and easily understood by problem-domain experts like teachers.

- Statistical methods and neural networks are deemed to be less suitable for data mining purposes. This rejection is due to the lack of comprehensibility. Knowledge models obtained under these paradigms are usually considered to be black-box mechanisms, able to attain very good accuracy rates but very difficult for people to understand. However, some of the algorithms of this type obtain models people can understand easily. For example, ADLinear, PolQuadraticLMS, Kernel and NNEP algorithms obtain functions that express the possible strong interactions among the variables.

Finally, in our educational problem the final objective of using a classification model is to show the instructor interesting information about student classification (prediction of marks) depending on the usage of Moodle courses. Then, the instructor could use this discovered knowledge for decision making and for classifying new students. Some of the rules discovered show that the number of quizzes passed in Moodle was the main determiner of the final marks, but there are some others that could help the teacher to decide whether to promote the use of some activities to obtain higher marks, or on the contrary, to decide to eliminate some activities because they are related to low marks. It could be also possible for the teacher to detect new students with learning problems in time (students classified as FAIL). The teacher could use the classification model in order to classify new students and detect in time if they will have learning problems (students classified as FAIL) or not (students classified as GOOD or EXCELLENT).

5 Conclusions

In this paper we have compared the performance and usefulness of different data mining techniques for classifying students using a Moodle mining tool. We have shown that some algorithms improve their classification performance when we apply such preprocessing tasks as discretization and rebalancing data, but others do not. We have also indicated that a good classifier model has to be both accurate and comprehensible for instructors. In future experiments, we want to measure the compressibility of each classification model and use data with more information about the students (i.e. profile and curriculum) and of higher quality (complete data about students that have done all the course activities). In this way we could measure how the quantity and quality of the data can affect the performance of the algorithms. Finally, we want also test the use of the tool by teachers in real pedagogical situations in order to prove on its acceptability.

Acknowledgments. The authors gratefully acknowledge the financial subsidy provided by the Spanish Department of Research under TIN2005-08386-C05-02 projects. FEDER also provided additional funding.

References

[1] Alcalá, J., Sánchez, L., García, S., del Jesus, M. et. al. KEEL A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems. Soft Computing, 2007.

[2] Baker, R., Corbett, A., Koedinger, K. Detecting Student Misuse of Intelligent Tutoring Systems. Intelligent Tutoring Systems. Alagoas, 2004. pp.531–540.

[3] Barandela, R., Sánchez, J.S., García, V., Rangel, E. Strategies for Learning in Class Imbalance Problems. Pattern Recognition 2003, 36(3), pp.849-851.

[4] Breiman, L. Friedman, J.H., Olshen, R.A., Stone, C.J. Classification and Regression Trees. Chapman & Hall, New York, 1984.

[5] Broomhead, D.S., Lowe, D. Multivariable Functional Interpolation and Adaptative Networks. Complex Systems 11, 1988, pp.321-355.

[6] Chen, G., Liu, C., Ou, K., Liu, B. Discovering Decision Knowledge from Web Log Portfolio for Managing Classroom Processes by Applying Decision Tree and Data Cube Technology. Journal of Educational Computing Research 2000, 23(3), pp.305–332.

[7] Chi, Z., Yan, H., Pham, T. Fuzzy Algorithms: with Applications to Image Processing and Pattern Recognition. World Scientific, Singapore, 1996.

[8] Clark, P., Niblett, T. The CN2 Induction Algorithm. Machine Learning 1989, 3(4), pp.261-283

[9] Cocea, M., Weibelzahl, S. Can Log Files Analysis Estimate Learners' Level of Motivation? Workshop on Adaptivity and User Modeling in Interactive Systems, Hildesheim, 2006. pp.32-35.

[10] Corcoran, A.L., Sen, S. Using Real-valued Genetic Algorithms to Evolve Rule Sets for Classification. Conference on Evolutionary Computation, Orlando, 1994, pp.120-124.

[11] Delgado, M., Gibaja, E., Pegalajar, M.C., Pérez, O. Predicting Students' Marks from Moodle Logs using Neural Network Models. Current Developments in Technology-Assisted Education, Badajoz, 2006. pp.586-590.

[12] del Jesus, M.J., Hoffmann, F., Junco, L., Sánchez, L. Induction of Fuzzy-Rule-Based Classifiers With Evolutionary Boosting Algorithms. IEEE Transactions on Fuzzy Systems 2004, 12(3), pp.296-308.

[13] Dougherty, J., Kohavi, M., Sahami, M. Supervised and Unsupervised Discretization of Continuous Features. Conf. on Machine Learning, San Francisco, 1995. pp.194–202.

[14] Fausett, L., Elwasif, W. Predicting Performance from Test Scores using Backpropagation and Counterpropagation. IEEE Congress on Computational Intelligence, 1994. pp.3398–3402.

[15] Espejo, P.G., Romero, C., Ventura, S. Hervas, C. Induction of Classification Rules with Grammar-Based Genetic Programming. Conference on Machine Intelligence, 2005. pp.596-601.

[16] Hämäläinen, W., Vinni, M. Comparison of machine learning methods for intelligent tutoring systems. Conference Intelligent Tutoring Systems, Taiwan, 2006. pp. 525–534.

[17] Jovanoski, V., Lavrac, N. Classification Rule Learning with APRIORI-C. In: Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving, 2001. pp.44-51.

[18] Kotsiantis, S.B., Pintelas, P.E. Predicting Students Marks in Hellenic Open University. Conference on Advanced Learning Technologies. IEEE, 2005. pp.664- 668.

[19] Martínez, D. Predicting Student Outcomes Using Discriminant Function Analysis. Annual Meeting of the Research and Planning Group. California, 2001. pp.163-173.

[20] Martínez, F.J., Hervás, C., Gutiérrez, P.A., Martínez, A.C., Ventura, S. Evolutionary Product-Unit Neural Networks for Classification. Conference on Intelligent Data Engineering and Automated Learning. 2006. pp.1320-1328.

[21] Minaei-Bidgoli, B., Punch, W. Using Genetic Algorithms for Data Mining Optimization in an Educational Web-based System. Genetic and Evolutionary Computation, Part II. 2003. pp.2252–2263.

[22] Moller, M.F. A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Neural Networks, 1993, 6(4), pp.525-533.

[23] Otero, J., Sánchez, L. Induction of Descriptive Fuzzy Classifiers with the Logitboost Algorithm. Soft Computing 2005, 10(9), pp.825-835.

[24] Platt, J. A Resource Allocating Network for Function Interpolation. Neural Computation, 1991, 3(2), pp.213-225.

[25] Quinlan, J.R. C4.5: Programs for Machine Learning. Morgan Kaufman. 1993.

[26] Romero, C., Ventura, S. Educational Data Mining: a Survey from 1995 to 2005. Expert Systems with Applications, 2007, 33(1), pp.135-146.

[27] Rustagi, J.S.: Optimization Techniques in Statistics. Academic Press, 1994.

[28] Sánchez, L., Couso, I., Corrales, J.A. Combining GP Operators with SA Search to Evolve Fuzzy Rule Based Classifiers. Information Sciences, 2001, 136(1-4), pp.175-191.

[29] Sánchez, L., Otero, J. Boosting Fuzzy Rules in Classification Problems under Single-winner Inference (in press). International Journal of Intelligent Systems. 2007.

[30] Superby, J.F., Vandamme, J.P., Meskens, N. Determination of Factors Influencing the Achievement of the First-year University Students using Data Mining Methods. Workshop on Educational Data Mining, 2006. pp.37-44.

[31] Venturini, G. SIA A Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts. Conf. on Machine Learning, 1993. pp.280-296.

[32] Wilson, S.W. Classifier Fitness Based on Accuracy. Evolutionary Computation, 1995, 3(2), pp.149-175.

[33] Yao, X. Evolving Artificial Neural Networks. Proceedings of the IEEE, 1999, 87(9), pp.1423-1447.

[34] Yudelson, M.V., Medvedeva, O., Legowski, E., Castine, M., Jukic, D., Rebecca, C. Mining Student Learning Data to Develop High Level Pedagogic Strategy in a Medical ITS. AAAI Workshop on Educational Data Mining, 2006. pp.1-8.

Acquiring Background Knowledge for Intelligent Tutoring Systems

Cláudia Antunes [1]
claudia.antunes@ist.utl.pt
[1] Dep of Computer Science and Engineering,
Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001, Lisboa, Portugal

Abstract. One of the unresolved problems faced in the construction of intelligent tutoring systems is the acquisition of background knowledge, either for the specification of the teaching strategy, or for the construction of the student model, identifying the deviations of students' behavior. In this paper, we argue that the use of sequential pattern mining and constraint relaxations can be used to automatically acquire that knowledge. We show that the methodology of constrained pattern mining used can solve this problem in a way that is difficult to achieve with other approaches.

1 Introduction

In the last years, the construction of intelligent tutoring systems has changed, shifting from a story boarding paradigm to a construction based on the separation of the content and instructional methods [6]. Along with the use of this new paradigm, the explicit and modular representation of the knowledge appeared as a fundamental task, with its acquisition being a central problem.

In order to solve this problem, in the last years, several tools were developed to automatically extract the background knowledge from data. Among the machine learning techniques used in those tools, the unsupervised ones have appeared as the most promising ones [9].

In this paper, we argue that sequential pattern mining is one of the unsupervised techniques of interest to this task, and, in conjunction with constraints and constraint relaxations, it can be used to discover the usual misconceptions or the bug library. A methodology for performing this discovery based on the use of sequential pattern mining and constraint relaxations is proposed and analyzed in this context.

The remaining of this paper is structured as follows: after presenting the sequential pattern mining problem and its solutions, the new methodology is described. This description is followed by a case study, where curricula instantiations are found. The paper concludes by presenting the conclusions and some guidelines for future work.

2 Sequential Pattern Mining

In general, it is possible to distinguish two major classes among unsupervised learning tasks: clustering and pattern mining. While clustering tries to identify groups of elements, that are similar within each group and dissimilar between groups, pattern mining aims at discover the set of "behaviours" that occur in the data a significant number of times. One

of the interesting characteristics of pattern mining is that with a few additional features, pattern mining algorithms are able to discover structured behaviours, and, in particular sequential patterns. These patterns exist when the data to be mined has some sequential nature, i.e., when each piece of data is an ordered set of elements. An example of such data is the sequence of actions performed and results achieved by some student, when interacting with a learning environment.

Sequential Pattern Mining is the machine learning task that addresses the problem of discovering the existing frequent sequences in a given database. The problem was first introduced in 1995 [1], and can be specified as

> *Given a set of sequences and some user-specified minimum support threshold σ, the goal of sequential pattern mining is to discover the set of sequences that are contained in at least σ sequences in the dataset, this is the set of frequent sequences.*

At this point it is important to make some remarks: first, a sequence is an ordered set of itemsets; second, an itemset is a non-empty set of items of interest for the analysis (the itemset composed of two items a and b is represented by (a,b)); finally, a sequence s is contained in another sequence t if all the itemsets of s are contained in some itemset of t, preserving the original order of occurrence.

2.1 Main Drawbacks and the Use of Constraints

In the last years, several sequential pattern mining algorithms were proposed, like *GSP* [9] and *PrefixSpan* [8]. In general, these algorithms act incrementally, discovering a pattern with k items, after the discovery of patterns with (k-1) items. This approach reduces the search space at each step, making use of the anti-monotonicity property. This property is related with the fact that a pattern can only be frequent if all its sub-patterns are also frequent. Despite the reasonable efficiency of pattern mining algorithms, the lack of focus and user control has hampered the generalized use of pattern mining. Indeed, the usually large number of discovered patterns makes the analysis of discovered information a difficult task. In order to solve this problem, several authors have promoted the use of constraints, where a constraint is defined as a predicate on the set of finite sequences. In this manner, given a database of sequences, some user-specified minimum support threshold σ and a constraint, a sequence is frequent if it is contained in at least σ sequences in the database and satisfies the constraint. This approach has been widely accepted by the data mining community, since it allows the user to control the mining process, either by introducing his background knowledge deeply into the process or by narrowing the scope of the discovered patterns. The use of constraints also reduces the search space, which contributes significantly to achieve better performance and scalability levels. When applied to sequential pattern mining, constraints over the content can be just a constraint over the items to consider, or a constraint over the sequence of items. More recently, regular languages have been proposed [4] and used to constrain the mining process, by accepting patterns that may be accepted by a regular language, represented as a deterministic finite automaton (DFA). Subsequent work [2] has shown that regular languages can be substituted by context-free languages, without compromising the performance of the algorithms. This is useful because context-free languages are more expressive than regular ones, and the only adaptation needed is the

substitution of the finite automaton by a pushdown automaton.

3 Acquisition of Background Knowledge for Student Modeling

One of the central components of intelligent tutoring systems is the student model, which is a qualitative representation that accounts for student behaviour in terms of existing background knowledge, and represents the system's belief about the learner's knowledge [11]. It comprises two distinct forms of knowledge: the domain theory and the bug library [9]. The domain theory corresponds to the ideal model of students' behaviour and in some cases it is completely specified. On the other side, the bug library collects the set of misconceptions held and other errors made by a population of students. If it is not impossible to enumerate all these facts, it is certainly very difficult to do it with the existing approaches.

Considering these aspects, we propose a new methodology to acquire background knowledge for student modelling. This methodology assumes that the curriculum knowledge can be represented by a context-free language and the bug library can be found by sequential pattern mining using constraint relaxations. The idea is that the curriculum knowledge can be used to guide the search of the facts in the bug library as a constraint, while the search is performed by a sequential pattern mining algorithm. Since constraints filter all the non-accepted sequences, a softer filter is needed. Constraint relaxations can perform the role of these softer filters, and can enable the discovery of patterns that deviate from the curriculum. Different classes of relaxations express the different levels of deviation of interest to the analyst.

3.1 Constraint Relaxations

The notion of constraint relaxation has been widely used when real-life problems are addressed. In sequential pattern mining they were first introduced in [4], where a regular expression was used to constrain the mining process, and some relaxations were used to improve the performance of the algorithm. A constraint relaxation can be seen as an approximation to the constraint. When used instead of the constraint, it enables the discovery of unknown information that will approximately match user expectations. If these relaxations are used to mine new patterns, instead of simply used to filter the patterns, the discovery of unknown information is possible [3].

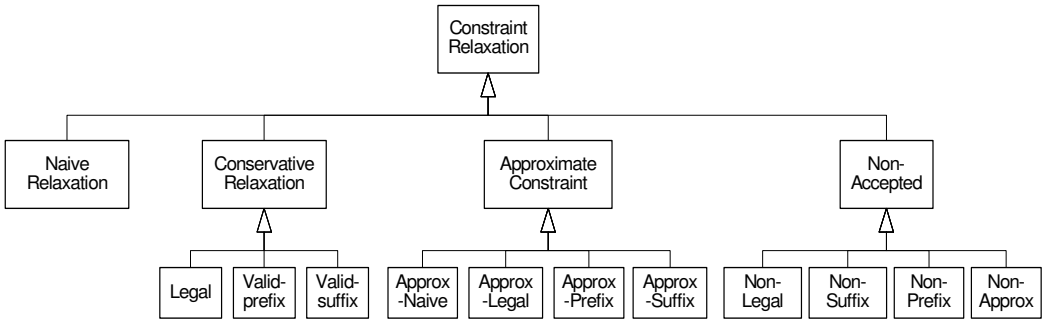


Figure 1. Hierarchy of constraint relaxations

Conservative relaxations group the already known relaxations, used in SPIRIT [4], and a third one – the *Valid-Prefix*. These relaxations impose a weaker condition than the original constraint, accepting patterns that are subsequences of accepted sequences. Although these relaxations have a considerable restrictive power, which improves significantly the focus on user expectations, they do not allow for the existence of errors.

Approx relaxations accept sequences that are approximately accepted by the constraint, which means that the patterns are at an acceptable edit distance from some sequence accepted by the constraint. This edit distance reflects the cost of operations (such as insertion deletion or replacement), that have to be applied to a given sequence, so it would be accepted as a positive example of a given formal language [5]. Approx relaxations can be combined with other relaxations resulting in a new set of relaxations. In general, approximate relaxations can be seen as less restrictive than conservative ones, and can be used to identify the common behaviours of students that made a limited number of errors when comparing to the curriculum knowledge.

The third class of relaxations is the *Naïve* relaxation, which corresponds to a simple item constraint. However, in the context of constraints expressed as formal languages, it can be seen as a relaxation that only accepts patterns containing the items that belong to the alphabet of the language. The naïve relaxation can be used to identify a portion of the frequent behaviours that comprise specific actions, permitting to understand the relations between those actions.

Finally, *Non-Accepted* relaxations accept patterns that are not accepted by the constraint. This type of relaxation is useful when there is a well-known model for the generality of sequences, and the goal is to identify the sequences that are not accepted by that model. In this manner, it is possible to identify low frequency behaviors that are still very significant to the domain. Fraud detection is the paradigm of such task. Note that the difficulties in fraud detection are related with the explosion of discovered information when the minimum support decreases.

It is important to note that the non-accepted relaxation will find all the patterns discovered by the other relaxations, representing a small improvement in the focus on user expectations. An interesting issue is to associate a subset of the alphabet in conjunction with non-accepted relaxation. This conjunction allows for focusing the mining process over a smaller part of the data, reducing the number of discovered sequences, and contributing to reach our goal. Like before, the sub-classes of Non-Accepted relaxations result by combining the non-acceptance philosophy with each one of the others relaxations. While non-accepted relaxation filters only a few patterns, when the constraint is very restrictive, the non-legal relaxation filters all the patterns that are non-legal with respect to the constraint. With this relaxation is possible to discover the behaviours that completely deviate from the accepted ones, helping to discover the fraudulent behaviours. A detailed definition of these relaxations for constraints specified as context-free languages see [3]. In the context of the acquisition of the bug library, the non-accepted relaxation is useful to discover the low-frequent misconceptions. Despite, they are less representative they could be very important as can be seen in the case study.

Table 1. List of common subjects, distributed by scientific area

Common subjects			
Mathematics	AL – Linear Algebra	**Programming Methodologies**	IP – Programming Introduction
	AM1 – Math. Analysis 1		AED – Algorithms and Data Structures
	AM2 – Math. Analysis 2		PLF – Logical and Functional Prog.
	AM3 – Math. Analysis 3		POO – Object Oriented Prog.
	PE – Probability and Statistics	**Architecture and Operating Systems**	SD – Digital Systems
	AN – Numerical Analysis		AC – Computer Architecture
Physics	FEX – Experimental Physics		SO – Operating Systems
	F1 – Physics 1	**Artificial Intelligence**	IA – Artificial Intelligence
	F2 – Physics 2	**Information Systems**	SIBD – IS and Databases
Computer Science	TC – Theory of Computation	**Computer Graphics**	CG – Computer Graphics

4 Curricula Analysis: a case study

In order to demonstrate our claims, consider the curriculum of an undergraduate program on information technology and computer science, with duration of five years (10 semesters) with 20 obligatory subjects (listed in Table 1), 16 subjects from a specific specialty area, an undergraduate thesis and 4 optional subjects in the last year. Also, consider there are four specialty areas: PSI – Programming and Information Systems; SCO – Computer Systems; IAR – Artificial Intelligence and IIN – Information Systems for Factory Automation. This information is usually publicly and previously known, and can be represented as a deterministic finite automaton (as shown on Figure 2). (This DFA shows the curriculum model for each specialty area (from the top to bottom: PSI, SCO, IAR and IIN, respectively; the existence of two different transitions per semester for SCO students, are due to a minor reorganization of the SCO curriculum on 1995/1996).

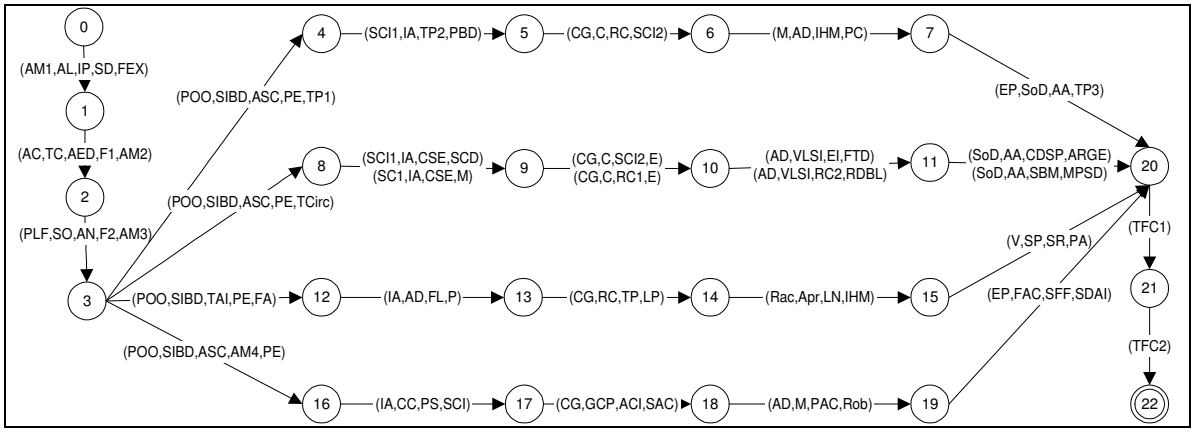


Figure 2. DFA for specifying the model curriculum for LEIC specialty areas

The data used in this study refers to the data of the undergraduate students of the Licenciatura em Engenharia Informática e de Computadores (LEIC) at Instituto Superior Técnico, from the academic years of 1989/90 to 1999/2000. From these students we have only considered the students that have registered for at least eight semesters, and therefore may have concluded the 4th year. In this manner, the dataset is composed of

1440 sequences, corresponding to the curriculum followed by each LEIC student. These sequences have an average length equal to 11.58 semesters. Most of the students (72%) have between 8 and 12 enrolments. In terms of the number of enrolments per semester, its mean is 4.82 enrolments on subjects per semester, with most of students (75%) enrolling on between 4 and 6 units.

Another interesting issue is the distribution of students per specialty area: 55% are PSI students, 19% SCO students, and IAR and IIN have 13% of students each. This distribution conditions the number of enrolments per course. For example, subjects exclusive to Artificial Intelligence and IIN have at most 13% of support. It is interesting to note that only 823 students (57%) have concluded the undergraduate thesis (TFC1 and TFC2). Since it is usual that students only took optional subjects in parallel or after finishing the undergraduate thesis, the support for optional subjects is at most 57%. Since the options are chosen from a large set of choices (130 subjects), their individual support is considerably lower. Indeed the course on Management (G) is the optional course with more students, about 40%.

The goal of this study is to demonstrate that with the use of the program curriculum as background knowledge and the use of constraint relaxations is possible to discover the frequent students' behaviours that approximately follow the curriculum. Note that if the background knowledge is used as a constraint to the sequential pattern mining process, only five patterns can be discovered: one for each specialty area (two for SCO). Moreover, it is probable that each pattern would have very low supports, since just a few students conclude all the subjects on their first enrolment. Next, we will present two experiments that illustrate the utility and effectiveness of our approach, respectively.

4.1 Finding frequent behaviours per specialty area

The first problem is related to the discovery of frequent behaviours per specialty area, and demonstrates the utility of our methodology. It is non trivial due to the difficulty of discovering which optional subjects are frequently chosen by which students. The difficulty of this task resides on the fact that all non-common subjects can be chosen as optional by some student. In this manner, a simple count of each subject support does not give the expected answer, since most of the subjects are required to some percentage of students.

The other usual approach to find out the frequent choices would be to query the database to count the support of each course, knowing that students have followed some given curriculum. However, this approach is also unable to answer the question, since a considerable number of students (more than 50%) have failed one or more subjects, following a slightly different curriculum.

In order to discover frequent behaviours, we have used the new methodology with the constraint based on the DFA in Figure 3, which corresponds to a sub-graph of the previous one. This automaton accepts sequences that represent the curricula on the fourth curricular year for each specialty area. With a constraint defined over this DFA filters all patterns that do not respect the model curriculum for the last two curricular years.

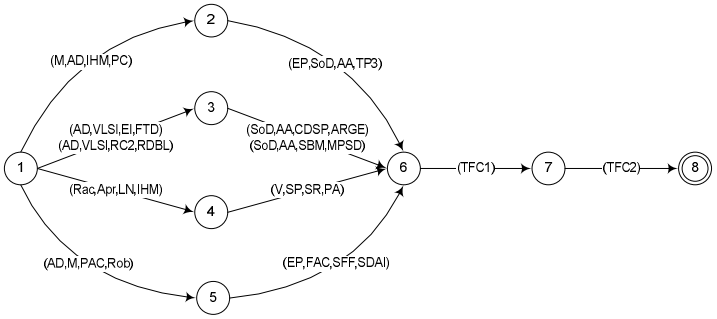


Figure 3. DFA for constraining the search of frequent behaviors per specialty areas

The use of constrained sequential pattern mining (with the specified constraint) would not contribute significantly to answer the initial question, since it would only achieve results similar to the ones achieved by the query described above. However, the use of the *Approx* relaxation that accepts at most two errors (ε=2) chosen from a restricted alphabet composed by every non-common course, allows the discovery of 25 patterns with a support at least equal to 1% (Table 2). These patterns show that, in general, students mostly attend Computer Graphics and Economical subjects.

Table 2. Some of the discovered patterns with optional subjects

Specialty Area	Curricula Instantiations
PSI	(M,AD,IHM,PC)(AA,SoD,EP,TP3)(TFC1,PAC)(TFC2,GF)
	(M,AD,IHM,PC)(AA,SoD,EP,TP3)(TFC1,Econ)(TFC2,GF)
	(M,AD,IHM,PC)(AA,SoD,EP,TP3)(TFC1,Econ)(TFC2,IG)
	(M,AD,IHM,PC)(AA,SoD,EP)(TFC1)(TFC2,TE2)
IIN	(M,AD,PAC,Rob)(EP,SDAI,SFF)(TFC1)(TFC2,IG)
	(M,AD,PAC,Rob)(EP,FAC,SDAI,SFF)(TFC1,IHM)(TFC2,GF)
	(M,PAC,Rob)(EP,FAC,SDAI,SFF)(TFC1,G)(TFC2)
	(M,PAC,Rob)(EP,FAC,SDAI,SFF)(TFC1,TE1)(TFC2)
IAR	(IHM,Rac,LN)(SP,V,PA,SR)(TFC1,TE1)(TFC2)
	(IHM,A,Rac,LN)(SP,V,PA,SR)(TFC1)(TFC2,GF)
SCO	(C)(VLSI,RC2,RDBL,AD)(AA,CDPSD,ARGE,SoD)(TFC1,G)(TFC2)
	(Elect)(VLSI,RC2,RDBL,AD)(AA,CDPSD,ARGE,SoD)(TFC1,G)(TFC2)

It is interesting to note that whenever IIN students have failed on some course on the 4th year, they choose one of two particular courses in Economy; the same is true for PSI and IAR students (shadowed patterns in Table 2). Note that in order to discover these rules, we have to be able to admit some errors on the obligatory curricula per specialty area, which is not easily achieved by executing a query to a database.

4.2 Finding Abandon Reasons

The great difficulty in determining the effectiveness of our mining process is to determine which patterns are the relevant or interesting ones. In order to perform this evaluation, we considered a smaller problem whose results can be easily analyzed. The selected problem is to find the reasons why students abandon LEIC before concluding the

42 subjects required. This problem was chosen because it is easy to enumerate some reasons for abandon in LEIC. By applying common sense, we can suggest two different reasons: the inability to conclude the first computer science specific subjects ('Programming Introduction'-[IP], 'Digital Systems'-[SD], 'Algorithms and Data Structures'-[AED] and 'Computer Architecture'-[AC]), and the inability to conclude the generic engineering subjects ('Linear Algebra'-[AL] and 'Mathematical Analysis 1 and 2'-[AM1, AM2]). This knowledge can be represented by the automaton in Figure 4.

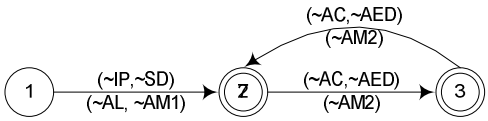


Figure 4. DFA for specifying the anticipated abandon reasons

The dataset used to analyze this question consists on the set of sequences corresponding to the curriculum followed by each LEIC student, with his first enrolment made between 1989 and 1997. The dataset includes all the students that abandoned LEIC (at least temporarily). The dataset (LEICabandons) contains 489 sequences.

In order to choose the relevant patterns, we applied the pattern discovery algorithms on both the LEICabandons and LEIC1989-2001 datasets, and identified as relevant the sequences that are frequent in the first dataset but are not frequent in the second one. With a support of 50%, 91 sequences have been discovered. From these, 79 are relevant by this criterion. A simple analysis of those sequences shows that most students that cancel their registration are not able to conclude more than two subjects in the second semester. Additionally, this analysis shows that the cancellation reasons anticipated and represented in the automaton in Figure 4 are close to the real reasons.

Table 3. Precision and Recall

	Unconst.	Accepted	Legal	Approx (ε=1)	Approx (ε=2)	Approx (ε=3)	Non-Acc
Nr Total Retrieved	91	2	15	20	71	90	89
Nr Retrieved and Relevant	79	2	10	18	63	78	77
Recall	100%	3%	13%	23%	80%	99%	97%
Precision	87%	100%	67%	90%	89%	87%	87%

Assuming these sequences are relevant for this task, it is now possible to evaluate the effectiveness of the new methodology by comparing its results with the results reached with constraints, by counting the number of relevant sequences discovered with each relaxation. Considering the notions of precision and recall usually used for evaluating the effectiveness of information retrieval algorithms, it is possible to compare the use of constraints with the use of relaxations as proposed in this work. When applied to the mining process, precision corresponds to the ratio of relevant patterns retrieved by the process to all patterns retrieved by the process, and recall corresponds to the ratio of relevant patterns retrieved by the process to all relevant patterns in the dataset. Applying those measures it is clear that there are significant differences among the relaxations, as shown in Figure 5.

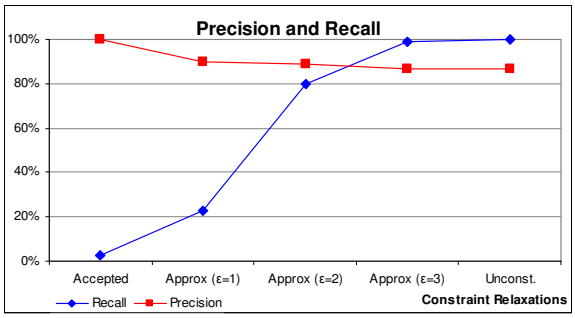


Figure 5. Precision and Recall chart

Since there are only two retrieved sequences when the constraint is used, it is clear that the existing background knowledge is not complete, but is correct. In this manner, the recall of the constrained process is usually very low. The existence of more accurate background knowledge would increase the recall. On the other side, all the accepted sequences are relevant, which makes the precision of the process 100%. The relaxation that shows a better balance between those measures and the efficiency of the process is the approx relaxation, because it is possible to adjust the number of patterns discovered without compromising the precision. By increasing or decreasing the number of errors allowed, it is possible to compensate the excessive selectiveness of the background knowledge. Note that the recall increases considerably when the number of allowed errors increase, but the decrease of the precision is less accentuated. From this analysis and the other experiments presented in this work, it is clear that the great challenge of pattern mining, in general, and of sequential pattern mining, in particular, is to reach the balance between the number of discovered patterns and the user's ability to analyze those patterns with the ability to discover unknown and useful information. The use of constraint relaxations, proposed in this work, represents a first step in this direction.

5 Conclusions

One of the problems in the construction of intelligent tutoring systems is the acquisition of background knowledge, both the teaching strategy and students' frequent behaviour. In this paper, we proposed a methodology to acquire parts of that knowledge, based on the use of sequential pattern mining. Our methodology consists of discovering the frequent sequential patterns among the recorded behaviours, keeping the discovery limited to the sequences that, in some manner, are approximately in accordance to the existing background knowledge. The methodology assumes that the existing background knowledge can be represented by a context-free language, which plays the role of a constraint in the sequential pattern mining process. The use of the constraint relaxations enables the discovery of unknown patterns that correspond to the deviations to the expected behaviours. Different classes of relaxations express different levels of deviation of interest to the analyst. In this paper, we have applied the methodology on identifying the deviations of students' behaviour from a pre-specified curriculum. In order to do that, we have represented the curriculum knowledge as a finite automaton, which establish the order of subjects that a student should attend to finish his graduation. Along with the use of sequential pattern mining, we tried the different relaxations, to answer specific

challenges ranging from the identification of the frequent curricula instantiations, to the discovery of abandon reasons. Although the identified behaviours can be used to classify a student and to predict his next result, alone they do not identify the causes of the failures. An interesting open issue is the correlation of the students' results with a specific teaching strategy. If this strategy and the corresponding expected results are known beforehand, our methodology can be used to identify the steps of the strategy that do not result as expected. In particular, the strategy in conjunction with the expected results can be represented as a context-free language and used as a constraint, to the sequential pattern mining process. In addition, an approx constraint will be able to discover the behaviour patterns that slightly deviate from the expected ones, which identify the failure steps.

References

[1] Agrawal, R and Srikant, R. Mining sequential patterns, *Proc Int'l Conf Data Engineering*, IEEE Computer Society Press, 1995, p. 3-14

[2] Antunes, C. and Oliveira, A.L. Inference of Sequential Association Rules Guided by Context-Free Grammars. *Proc. Int'l Conf. on Grammatical Inference*, 2002, p. 1-13

[3] Antunes, C. and Oliveira, A.L. Constraint Relaxations for Discovering Unknown Sequential Patterns, in B. Goethals and A. Siebes (Eds.): *KDID 2004*, LNCS 3377, Springer-Verlag Berlin Heidelberg, 2005, p.11–32

[4] Garofalakis, M., Rastogi, R. Shim, K. SPIRIT: Sequential Pattern Mining with Regular Expression Constraint. Proc. *Int'l Conf. Very Large Databases*, 1999, 223-234.

[5] Levenshtein, V., Binary Codes capable of correcting spurious insertions and deletions of ones". *Problems of Information Transmission*, 1965, vol.1 1, p. 8-17.

[6] Murray, T Expanding the Knowledge Acquisition Bottleneck for Intelligent Tutoring Systems". *Int'l Journal of Artificial Intelligence in Education*, 1997, vol. 8, p. 222-232.

[7] Murray, T Authoring Intelligence Tutoring Systems: an Analysis of the State of the Art". *Int'l Journal of Artificial Intelligence in Education* 1999, vol. 10, p. 98-129.

[8] Pei J, Han J, Mortazavi-Asl B et al., PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth". *Proc Int'l Conf Data Engineering*. IEEE Computer Society Press, 2001

[9] Sison, R, Shimura, M. Student Modeling and Machine Learning. *Int'l Journal of Artificial Intelligence in Education* 1998, vol. 9, p. 128-158.

[10] Srikant,R, Agrawal,R Mining Sequential Patterns: generalizations and performance improvements. *Proc Int'l Conf Extending Database Technology*. 1996, 3-17

[11] Stauffer, K. Student Modelling and Web-based Learning Systems – Technical Report, Athabasca University. 1996

Analytic Comparison of Three Methods to Evaluate Tutorial Behaviors

Jack Mostow and Xiaonan Zhang
mostow@cs.cmu.edu, xiaonanz@cs.cmu.edu
Project LISTEN, School of Computer Science, Carnegie Mellon University

Abstract. We compare the purposes, inputs, representations, and assumptions of three methods to evaluate the fine-grained interactions of intelligent tutors with their students. One method is conventional analysis of randomized controlled trials (RCTs). The second method is learning decomposition, which estimates the impact of each practice type as a parameter in an exponential learning curve. The third method is knowledge tracing, which estimates the impact of practice as a probability in a dynamic Bayes net. The comparison leads to a generalization of learning decomposition to account for slips and guesses.

1 Introduction

Educational data mining researchers have used various methods to analyze fine-grained tutor data in order to evaluate the effects of tutorial actions on student behavior, including randomized controlled trials (RCTs), learning decomposition, and knowledge tracing. Other papers [1, 2] compare these three methods empirically. In this paper we compare their fundamental characteristics descriptively and analytically. We now briefly describe these methods, and then the dimensions along which the remainder of the paper compares them. We describe the three methods in terms of the high-level strategic ideas they embody, the particular instantiations that we compare, and specific implementations of these instantiations.

1.1 Randomized controlled trials analysis

The high-level strategic idea of RCTs is that randomizing assignment to treatments allows the strong causal inference that significant differences in outcome are due to differences in treatment. In RCTs, participants are randomly assigned to receive one of several different interventions. RCT analysis aggregates and compares the outcomes under each condition, to assess the effectiveness of different interventions. For example, Project LISTEN's Reading Tutor [3] chooses randomly among multiple ways to help the student learn a given word. To compare their effectiveness using RCT analysis, the Reading Tutor randomly chooses for each student how to teach or practice a given word. For each such randomized trial, we measure its outcome, e.g., how well the student reads the word at the next encounter of it. By comparing the aggregated outcomes for each intervention, we can statistically estimate how effective they are relative to one another.

RCT analysis is instantiated by a statistical test or model to estimate the effects of treatment and possibly other variables. Common examples include t-test and ANOVA.

The test or model is implemented as the computation of a standard statistical formula, typically in a statistical package such as SPSS.

1.2 Learning decomposition

The strategic idea of learning decomposition is to distinguish among different types of exposure in fitting a model of learning to students' performance data, in order to obtain empirical estimates of the relative impact of different types of practice or instruction. Prior work [4-6] instantiates this idea in terms of a particular model form, namely an exponential learning curve, by expressing the amount of exposure as a function of the amount of exposure of each type, namely a linear combination of them.

Alternative model forms would produce alternative instantiations. One plausible alternative to an exponential learning curve is a power law. Power laws are often used to fit performance data averaged over multiple participants, but exponential curves tend to fit individual performance data better than power law curves do [7]. A more complex alternative [8] is based on a richer theoretical model of learning and forgetting over time, but must be expressed recursively instead of in closed form, and has more parameters to fit.

The instantiation of learning decomposition used in this paper generalizes the classic exponential learning curve to distinguish the effectiveness of *m* different types of practice, as shown in Equation 1:

$$performance = A \times e^{-b \times (\beta_1 t_1 + \ldots + \beta_m t_m)} \qquad \textbf{(1)}$$

In the equation, *performance* measures a learning outcome, such as error rate, help requests, or response time: the lower the value, the better the student learned. The variables t_i $(i=1\ldots m)$ represent the amount of type *i* practice that the student has had, as measured by the number of practice opportunities of that type. The free parameters *A*, *b*, and β_i's are estimated from observations of student performance. Here *A* represents the student's initial performance without any prior practice, and *b* is the learning rate. Finally, the free parameters β_i $(i=2\ldots m)$ represent the impact of type *i* practice relative to type 1 practice, whose impact β_1 we define to be 1 as a baseline for comparison.

For example, consider how learning decomposition can apply to our previous example of comparing different tutorial interventions to teach student a word. The *performance* measure can be word reading time. The variable t_i counts the number of times the student received intervention of type *i*. The estimate of each β_i measures the effectiveness of intervention type *i* compared to the baseline. For instance, if $\beta_2 = 2$, it means that on average a type 2 intervention is twice as helpful as a type 1 intervention.

Implementation of learning decomposition refers to how the model is fit to the data. For example, our implementation uses SPSS to perform unconstrained non-linear regression using the Levenberg-Marquardt procedure.

1.3 Knowledge tracing

The general idea of knowledge tracing is to model a student's performance and changing knowledge state during skill acquisition, and to use the model to update an estimate of

this state based on successive observations of student performance. Atkinson [9] proposed a Markov model relating knowledge to performance. Corbett and Anderson [10] used a simple 2-state model – the state of not knowing a given skill, and the state of knowing the skill.

An instantiation of knowledge tracing specifies a particular model of learning. We represent the two knowledge states in Corbett and Anderson's model as a hidden node in a dynamic Bayes network (DBN). The value of this node is true if the student knows the skill at a given step, and false if not. The node is hidden, so its value cannot be observed directly. Instead, knowledge tracing maintains the probability Pr(K_i) that the student has learned that skill at time step *i*. It uses observations of the student's performance to update this probability each time the student encounters an opportunity to use the skill. We extend this basic knowledge tracing model to incorporate and compare the efficacy of different types of tutorial interventions. Figure 1 shows the graphical representation of the extended model.

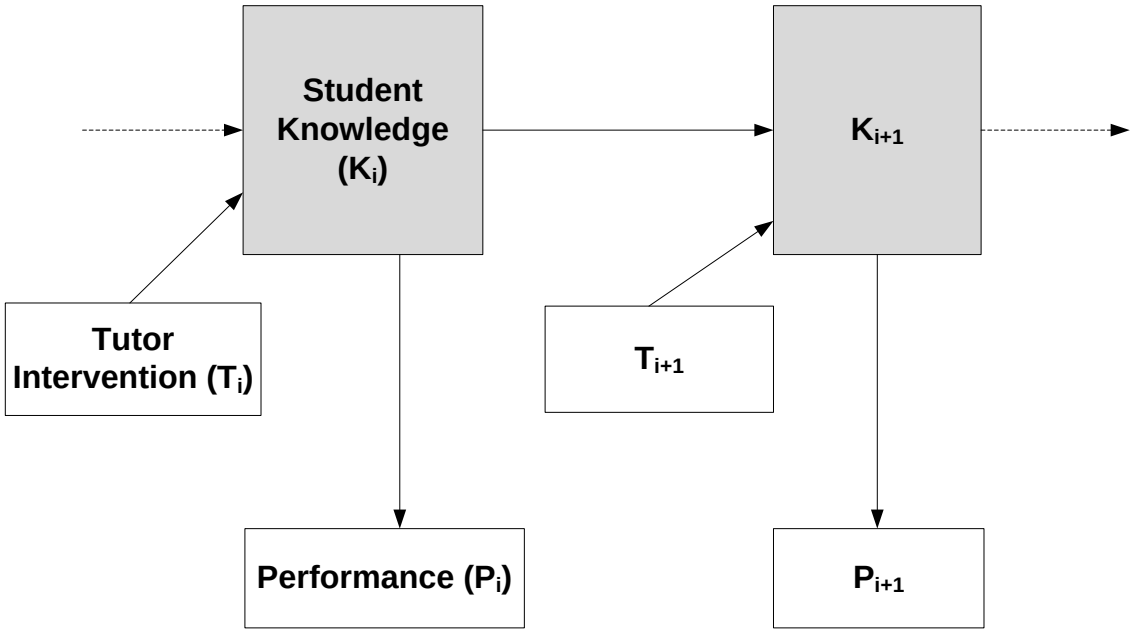


Figure 1. Knowledge tracing model extended with "Tutor Intervention" node

From the graph we can see that the model is a DBN with 3 nodes at each time step *i*. The binary valued hidden node K_i represents whether the student knows the skill at time *i*. Pr(K_i) is a probabilistic estimate of this binary student knowledge variable. P_i represents the student's observed performance (e.g. correct or incorrect) on that skill at time *i*. Using a DBN for knowledge tracing lets us generalize its original formulation [10] by including additional nodes. For instance, Chang et al. [11] added a node to represent help requests. To evaluate interventions, we include tutor intervention T_i as a discrete variable to represent which tutorial intervention is made at time *i*. Links in the model encode conditional dependencies between variables.

A model is described by the following parameters:

- *knew*: Probability that the student already knew the skill prior to any instruction
- *prob_Ti*: Probability that the intervention is of type *i*.
- *learn_Ti*: Probability of acquiring the skill from an intervention of type T_i, i.e. P(K_{i+1}=true | K_i= false, T_i)
- *forget_Ti*: Probability of losing a known skill conditioned on an intervention of type T_i, i.e. P(K_{i+1}=false | K_i= true, T_i)
- *guess*: Probability of answering correctly without knowing the skill, i.e. P(P_i=true| K_i= false)
- *slip*: Probability of answering incorrectly despite knowing the skill, i.e., P(P_i=false| K_i= true).

The problem of comparing the effects of different types of tutorial behaviors then reduces to comparing the values of the *learn_Ti* and *forget_Ti* parameters for the different T_i's corresponding to those types of behaviors.

An implementation of this model includes an off-line training procedure to learn the model parameters from a corpus of performance data, and a runtime update procedure to revise the estimate of the student's knowledge based on observed performance at step *n*. The BNT-SM implementation of knowledge tracing [12] uses BNT's EM procedure to fit the model. The update procedure can be implemented by a general Bayes net inference procedure, or by a procedure to compute specific formulas derived from the model.

1.4 Dimensions of comparison

Basic questions about RCTs, learning decomposition, and knowledge tracing include:

- What outputs are these methods designed to produce?
- What inputs do they require?
- What models are they based on?
- What are their underlying assumptions?

Sections 2-5, respectively, compare the three methods with respect to these questions.

2 Intended output

The three methods differ in the purpose they are designed to achieve, that is, what outputs they compute.

RCT analysis compares trial outcomes to identify factors that predict or affect outcomes, primarily treatment condition. Thus it is a natural way to compare interventions. It can

estimate student ability that affects outcomes by including student identity as a factor in a statistical test, but such assessment does not exploit the randomized treatment assignment. Thus RCT analysis evaluates interventions more than students. Moreover, this evaluation analyzes an entire set of trials simultaneously, so even if it estimates student ability as a factor, it is not designed to update this estimate incrementally based on sequential observations of student performance.

Learning decomposition can also fit student parameters such as initial performance and learning rate, but its primary purpose is to compare the relative impact of different types of practice. Its performance prediction for how a student will do at the N^{th} encounter of a skill depends only on how much prior practice of each type he or she got on that skill – not on how well he or she performed on it previously – unless encounters are treated as different types based on their immediate outcomes.

Learning decomposition could incorporate information about prior performance by treating successful and unsuccessful encounters as two different types of practice. This possibility might in fact be interesting to explore. However, it does not escape the fact that simply counting the number of encounters of different types ignores order and recency effects. Thus the resulting model would predict the same performance after 5 correct responses followed by 5 incorrect responses as it would predict after 5 incorrect responses followed by 5 correct responses. However, learning decomposition can model some recency effects by treating spaced and massed practice as different types [4].

Knowledge tracing is specifically designed to update, at each practice opportunity, a tutor's estimate of an individual student's mastery of the skill(s) it exercises. Conditioning this update on practice type makes it possible to evaluate how each type of practice affects learning. Based on the latest knowledge estimate (plus the slip and guess parameters), the tutor can predict how well the student is about to perform, and plan accordingly – for example by picking an easier problem to prevent a likely imminent failure.

Knowledge tracing makes a more informed prediction of performance than learning decomposition does, because the prediction reflects performance during prior practice, not just the amount of practice. For example, suppose the data set includes sequences of N-1 practice opportunities by two different students on the same skill, with the same sequence of practice types, but that one sequence consists of successes and the other sequence consists of failures. Knowledge tracing will predict higher performance after the successes than after the failures, because its prediction is conditioned on observed performance. In contrast, learning decomposition – unless it includes any student-specific parameters or distinguishes past encounters by performance -- will predict the same performance in both cases, because its prediction is based solely on the number and type of practice opportunities.

Knowledge tracing updates its estimate of student knowledge based on sequential observations of student performance, and is sensitive to their order – but not their timing. Learning decomposition can address this limitation to some extent by treating massed and spaced trials as different types of practice. Analogously, knowledge tracing can

condition the parameters *learn* and *forget* based on the time elapsed between successive encounters of a skill. For both methods, this *ad hoc* refinement models temporal effects as discrete, for example by classifying a student's successive encounters of a skill as occurring on the same or different days [4]. A more complex method [8] models continuous temporal effects based on a deeper theory of memory.

3 Information input

The three methods differ in what data they input, that is, which observations they consider.

RCT analysis compares only observed outcomes, so it simply ignores trials whose outcomes were unobserved, undefined, or masked. For example, a student's performance in reading a word is unobserved when the Reading Tutor reads it, undefined when the Reading Tutor gives unrequested help on the word before accepting it, and masked by recency effects if the student has seen the word earlier the same day.

Learning decomposition analyzes trials both as outcomes and as practice. It excludes trials as outcomes where performance is unobserved, undefined, or masked, but in each case the encounter counts as practice.

Knowledge tracing represents outcomes and practice as the same performance nodes in a dynamic Bayes net, and tolerates missing and partial observations. For example, credit for a word is undefined if masked by tutor-initiated help, but the word encounter still updates the estimated probability that the student has learned the word.

4 Model form: representation, computation, and extension

The three methods differ in how they are represented, applied, and extended.

RCT analysis has no hidden variables to estimate, no initialization conditions to be sensitive to, and no local minima to get stuck on. An appropriate statistical test to compare outcomes between different subsets of trials is practically instantaneous in a standard statistical package such as SPSS, even with many thousands of trials. Extending the analysis involves adding outcome variables to test, and features to disaggregate by or include as factors in statistical tests such as ANOVA.

Learning decomposition estimates parameters that represent the impact of each mode of practice in an exponential model of performance over time. Non-linear regression in SPSS 11.0 took only about one minute to fit hundreds of models (one for each word) to thousands of word encounters in our empirical comparison of evaluation methods [1]. Extending models means adding parameters to further distinguish types of practice or other influence on performance, e.g. effect of length on word reading.

Knowledge tracing, generalized to distinguish different practice types, estimates parameters that represent the impact of each type of practice on a hidden knowledge state in a dynamic Bayes net, and the effect of this knowledge on observed student

performance. BNT-SM [12] takes about 10 hours to train and evaluate models on a data set that learning decomposition takes only a minute to fit. Extending the model means adding nodes to represent observations, and links to represent hypothesized causal influences.

5 Underlying assumptions

RCT analysis treats trials as separate and does not attempt to model the causal effects of one trial on the outcome of another, such as diminishing effects of successive trials on the same skill, or transfer effects from practice on one skill to performance on another [13]. At most it accounts for statistical dependencies among related trials by using tests with the appropriate number of degrees of freedom. For example, instead of treating all trials on a story word as independent, we can average outcomes for each type of practice over all the students who received that type of practice on the word.

In contrast, both learning decomposition and knowledge tracing assume particular models of how performance improves with practice. On the surface, they look very different. Knowledge tracing can be expressed as a dynamic Bayes net that incrementally updates estimates of a hidden skill, while learning decomposition is expressed as a closed-form exponential function that predicts performance directly.

Although their surface form differs markedly, the mathematics of these two methods is closely related at a deeper level in a way that is useful to elucidate because it not only reveals their underlying similarities, but shows how to bridge some of their differences. In particular, starting from a Bayesian model of knowledge tracing we can derive learning and performance curves that we can relate to learning decomposition.

5.1 Derivation of learning decomposition from knowledge tracing

Knowledge tracing estimates the probability $\Pr(K_n)$ that the student knows a given skill at time step n, according to a dynamic Bayes net model. For brevity we will abbreviate $\Pr(K_n)$ as K_n. The parameters of this model include the *knew* probability K_0 that the student already knew the skill prior to instruction, the *learn* probability L of acquiring the skill from a step, the *forget* probability of losing a skill, the *guess* probability g of answering correctly without knowing the skill, and the *slip* probability s of answering incorrectly despite knowing the skill.

We are interested in how K_n changes over time according to this model. The probability of knowing the skill at step n given that the student did not know it at step $n-1$ is L. If we assume zero probability of forgetting, the probability $(1 - K_n)$ of *not* knowing the skill at step n is the probability of not knowing the skill at the previous step, times the probability of not learning it, or $(1 - K_{n-1}) \times (1 - L)$. Consequently the ratio $(1 - K_n) / (1 - K_{n-1})$ is the constant $(1 - L)$, and the probability $(1 - K_n)$ can be expressed in closed form as $(1 - K_0) \times (1 - L)^n$. That is, ignorance at step n requires not knowing the skill in advance, followed by not learning it.

This formula expresses the *prior* probability of (not) knowing the skill at time *n*; it is not conditioned on any observed student performance. Observed performance affects our estimate of student skills, but this effect is *evidentiary*, not *causal*: according to the model, student performance does not influence student knowledge. In short, *even without any observation of student performance*, we can still use the model to predict student knowledge.

We are also interested in the performance predicted by the model, specifically in the error rate expected at step *n*. The probability W_n of a wrong response at step *n* can be split into two cases – either not knowing and not guessing, or knowing but slipping. Consequently $W_n = (1 - K_n) \times (1- g) + K_n \times s$. Plugging in our closed form expression for $(1 - K_n)$ gives

$$W_n = (1 - K_0) \times (1 - L)^n \times (1- g) + (1 - (1 - K_0) \times (1 - L)^n) \times s$$

$$= [(1 - K_0) \times (1- g) - (1 - K_0) \times s] \times (1 - L)^n + s$$

$$= s + (1 - g - s) \times (1 - K_0) \times (1 - L)^n.$$

What if the learning probability varies with the type of step? We can generalize $(1 - L)^n$ to the product $(1 - L_1) \times \ldots \times (1 - L_n)$. If there are *m* types of steps, with t_i steps of type *i*, each with learning probability L_i, we rewrite this product as $(1-L_1)^{t_1} \times \ldots \times (1-L_m)^{t_m}$, and generalize the predicted error rate to:

$$s + (1 - g - s) \times (1 - K_0) \times (1 - L_1)^{t_1} \times \ldots \times (1 - L_m)^{t_m}$$

If $s = g = 0$, this formula reduces to $(1 - K_0) \times (1 - L_1)^{t_1} \times \ldots \times (1 - L_m)^{t_m}$, which relates to the learning decomposition formula $A \times e^{-b \times (\beta_1 t_1 + \ldots + \beta_m t_m)}$ as follows. The factors $(1 - L_1) \ldots (1 - L_m)$ correspond to the coefficients $\beta_1, \ldots, \beta_m$ that represent the relative value of different types of practice. More precisely, since $(1 - L_i)^{t_i} = \exp(\log(1-L_i) \times t_i)$, the expression $\log(1 - L_i)$ corresponds to β_1. The factor $(1 - K_0)$ corresponds to $A \times e^{-b}$.

This derivation reveals that we can extend learning decomposition to include slip rate *s* and guess rate *g* in the generalized formula $s + (1 - g - s) \times A \times e^{-b \times (\beta_1 t_1 + \ldots + \beta_m t_m)}$. The additive term *s* represents the asymptotic performance beyond which the error rate will not decrease even with unlimited practice.

However, it is important to point out that the generalized formula, at least in the form above, applies to the rate of errors or help requests, but not to response time. The reason is that the parameter *s* appears both as an additive term and in the factor $(1 - g - s)$. If the formula represents a time quantity, *s* must be expressed in some unit of time. But the *s* in $(1 - g - s)$ must be unit-free to be comparable with the constant *1* and the probability *g*. Since *s* cannot be both a temporal quantity and unit-free, the overall formula cannot represent a time quantity, and in fact must itself be unit-free.

Extending this generalized formula to predict time or other continuous measures of performance requires modeling how they are affected by slips and guesses. For example, if guessing takes negligible time, the guess parameter *g* already models the effect of

guesses on performance time: if $s = 0$, then $g = 0.5$ cuts predicted performance time in half. But if guesses take non-negligible time, modeling them requires extending the formula by interpolating it with guessing time. Likewise, modeling the effect of slips on time requires replacing the additive term s with slip time, e.g. the time to misread a known word. The extended form of the generalized formula therefore looks like this, where the expressions *slip time* and *guess time* depend on how they are modeled:

$$s \times (\textit{slip time}) + g \times (\textit{guess time}) + (1 - g - s) \times A \times e^{-b \times (\beta_1 t_1 + \ldots + \beta_m t_m)}$$

6 Contributions

This paper provides a descriptive and analytical comparison of three methods to evaluate tutorial behaviors: RCT analysis, learning decomposition, and knowledge tracing using DBNs. We compare their inputs, outputs, models, and assumptions. In particular, we elucidate the underlying mathematical relationship between knowledge tracing and learning decomposition, thereby showing how to generalize learning decomposition to incorporate slip and guess parameters. We hope that other researchers will find this paper useful in making informed choices of which method(s) to use to model and evaluate learning in tutors.

Acknowledgements

The research reported here was supported by the National Science Foundation under ITR/IERI Grant No. REC-0326153, by the Institute of Education Sciences, U.S. Department of Education, through Grant R305B070458 to Carnegie Mellon University, and by the Heinz Endowments. The opinions expressed are those of the authors and do not necessarily represent the views of the National Science Foundation, the Institute, the U.S. Department of Education, or the Heinz Endowments.

References (LISTEN publications are at www.cs.cmu.edu/~listen)

[1] Zhang, X., J. Mostow, and J.E. Beck. A Case Study Empirical Comparison of Three Methods to Evaluate Tutorial Behaviors. *9th International Conference on Intelligent Tutoring Systems*, 122-131. 2008. Montreal: Springer-Verlag.

[2] Beck, J.E., K.-m. Chang, J. Mostow, and A. Corbett. Does help help? Introducing the Bayesian Evaluation and Assessment methodology. *9th International Conference on Intelligent Tutoring Systems* 2008. Montreal.

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

[4] Beck, J.E. Using learning decomposition to analyze student fluency development. *ITS2006 Educational Data Mining Workshop*, 21-28. 2006. Jhongli, Taiwan.

[5] Beck, J.E. Does learner control affect learning? *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, 135-142. 2007. Los Angeles, CA: IOS Press.

[6] Beck, J.E. and J. Mostow. How who should practice: Using learning decomposition to evaluate the efficacy of different types of practice for different types of students. *9th International Conference on Intelligent Tutoring Systems* 2008. Montreal.

[7] Heathcote, A., S. Brown, and D.J.K. Mewhort. The Power Law Repealed:The Case for an Exponential Law of Practice. *Psychonomics Bulletin Review*, 2000: p. 185-207.

[8] Pavlik Jr., 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.

[9] Atkinson, R.C. Ingredients for a theory of instruction. *American Psychologist*, 1972. *27*(10): p. 921-931.

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

[11] Chang, K.-m., J.E. Beck, J. Mostow, and A. Corbett. Does Help Help? A Bayes Net Approach to Modeling Tutor Interventions. *AAAI2006 Workshop on Educational Data Mining* 2006. Boston, MA.

[12] Chang, K.-m., J. Beck, J. Mostow, and A. Corbett. A Bayes Net Toolkit for Student Modeling in Intelligent Tutoring Systems. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 104-113. 2006. Jhongli, Taiwan.

[13] Zhang, X., J. Mostow, and J.E. Beck. All in the (word) family: Using learning decomposition to estimate transfer between skills in a Reading Tutor that listens. *AIED2007 Educational Data Mining Workshop* 2007. Marina del Rey, CA.

Labeling Student Behavior Faster and More Precisely with Text Replays

Ryan S.J.d. Baker[1], Adriana M.J.A. de Carvalho
rsbaker@cmu.edu, dikajoazeirodebaker@gmail.com
[1] Human Computer Interaction Institute, Carnegie Mellon University

Abstract. We present text replays, a method for generating labels that can be used to train classifiers of student behavior. We use this method to label data as to whether students are gaming the system, within 20 intelligent tutor units on Algebra. Text replays are 2-6 times faster per label than previous methods for generating labels, such as quantitative field observations and screen replays; however, being able to generate classifiers on retrospective data at the coder's convenience (rather than being dependent on visits to schools) makes this method about 40 times faster than quantitative field observations. Text replays also give precise predictions of student behavior at multiple grain-sizes, allowing the use of both hierarchical classifiers such as Latent Response Models (LRMs), and non-hierarchical classifiers such as Decision Trees. Training using text replay data appears to lead to better classifiers: LRMs trained using text replay data achieve higher correlation and A' than LRMs trained using quantitative field observations; Decision Trees are more precise than LRMs at identifying exactly when the behavior occurs.

1 Introduction

In recent years, there has been increasing interest in classifying what type of behavior a student is engaging in as he or she uses interactive learning software. For instance, models that can classify whether a student is gaming the system (attempting to succeed in an interactive learning environment by exploiting properties of the system rather than by learning the material [cf. 5]) have been developed for several learning systems [1, 6, 9, 20]. Models that can classify other types of behavior, such as help avoidance [1] and deep and shallow exploration behaviors [2] have also been developed. These models have been used as inputs to data mining analyses of student learning and motivation [8, 12], and have been used to select pedagogical interventions [3, 4, 20].

However, developing classifiers of student behavior has thus far been either highly time-consuming or has resulted in classifiers that are difficult to validate. In this paper, we present a method for quickly and accurately labeling data in terms of student behavior – text replays. The entire process of creating a detector for a unit in an intelligent tutor is about 40 times faster when text replays are used, compared to prior data labeling methods such as quantitative field observations or screen replays. Text replays also produce data which can be more easily used with off-the-shelf classification algorithms, such as those in Weka [21], than field observations. We validate this method by using it to develop classifiers of gaming the system for 20 different Cognitive Tutor lessons on algebra, and study those detectors' ability to predict which students game, how much they game, and when they game.

2 Prior Work

Several approaches have previously been used to develop classifiers of student behaviors, such as gaming the system, in interactive learning software.

One approach to developing classifiers of student behaviors has been to conduct quantitative field observations [5,17] of student behavior in the classroom (or other naturalistic settings), and then use classification algorithms on the resultant data [cf. 6, 20]. However, there are two drawbacks to this approach: First, this method is highly time-consuming, as it requires coders to travel to the setting of use and watch student behavior in real-time. Baker et al. [5] report making 563 classifications in around 7.5 hours of class time. However, since the observations in Baker et al. were made in classrooms distant from the researchers' offices, driving time and setup time should also be counted as part of the overall time cost of conducting these observations. If we assume around a half hour driving time in each direction, and 15 minutes setup time before each session, that works out to around 6 hours logistical time. Hence, while each observation took 20 seconds to conduct, the actual time cost per classification is 1.3 minutes. Additionally, this measure does not account for the difficulty of scheduling classroom observations within schools, and time lost to waiting for observation opportunities. Second, the resulting data labels are not immediately synchronized with the log data. Whereas log data is generally organized either at the semantic level of student actions (i.e. entering an answer, requesting help, etc.) or keystrokes/mouse movements [cf. 11]), the observations occupy 20-30 second time windows. In many cases, it is difficult to exactly synchronize these two types of data, since even a slight skew in time recording mechanisms may significantly affect accuracy. To compensate, researchers have probabilistically assigned time windows [cf. 20] or used hierarchical models that fit to overall proportions of behavior rather than labeling individual actions [cf.6].

A second approach to developing classifiers of student behavior is to use knowledge engineering rather than machine learning on labeled data [cf.1,9]. This approach is generally quite quick to implement, but it is not possible to directly validate the accuracy of classifications without obtaining labeled data of some sort. The single example of a validation of a knowledge engineering behavior classifier in a learning environment (that we are aware of) involved the use of data collected through quantitative field observations [cf. 18]. A related approach is to use clustering to develop behavior categories, and then develop a classifier based on those behavior categories [cf. 2]. Like knowledge engineering, this approach is quick to implement, but suffers from the same difficulty in model validation.

An alternate approach is to directly label the log files with the construct(s) of interest. One method for doing so, proposed by de Vicente and Pain [11], is screen replays – labeling a student's behavior or affect while watching a video capture of the student's screen during learning software usage. This approach avoids synchronization problems in labeling data, as well as scheduling/driving time, but is still quite time-consuming, as coders watch student behavior in real time. de Vicente and Pain [11] report observers making 85 classifications in around 6 hours, an average of one classification every 4.2

minutes. Cocea and Weibelzahl [10] also report having human coders label log files, although they do not report the coding process in full detail.

3 Text Replays

We propose another method for directly labeling log files with the constructs of interest: text replays. Text replays represent a segment of student behavior from the log files in a textual ("pretty-printed") form. A sequence of actions of a pre-selected duration (in terms of time or length) is shown in a textual format that gives information about the actions and their context. In the example shown in Figure 1, the coder sees each action's time (relative to the first action in the clip), the problem context, the input entered, the relevant skill (production), and how the system assessed the action (correct, incorrect, a help request, or a "bug"/ misconception). The coder can then choose one of a set of behavior categories (in this study, gaming or not gaming), or indicate that something has gone wrong, making the clip uncodeable.

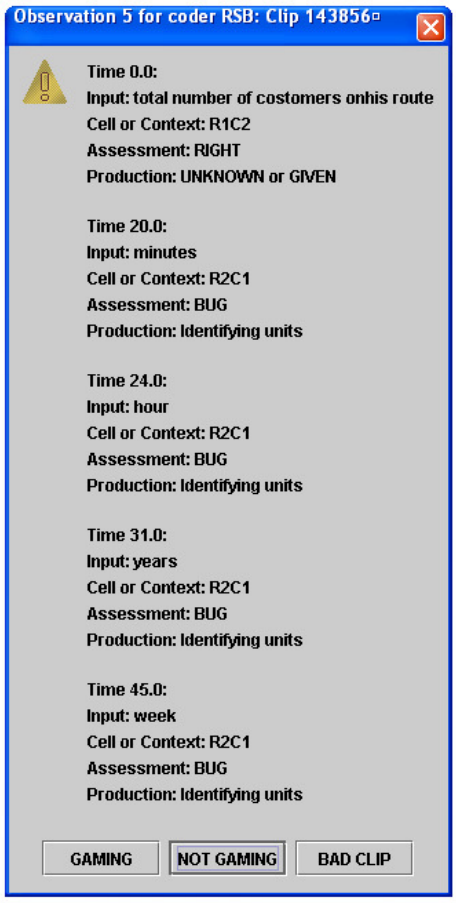


Figure 1: A "text replay" of student gaming

Text replays give relatively limited information, compared to quantitative field observations or full screen replays, omitting the visual information that would be seen in those methods (including mouse movements and partial responses). It is also necessary for the coder to understand the user interface in order to interpret contextual information. However, text replays are likely to be very quick to classify, and can be generated automatically from existing log files.

Text replays were first proposed in Baker, Corbett, & Wagner [7], where it was shown that text replays have good inter-rater reliability, and that text replays agree well with predictions made by models generated using data from quantitative field observations. In this paper, we test this technique's speed for labeling large amounts of data, develop classifiers of student behavior, and test these classifiers' goodness of fit.

4 Text Replay Labeling Study

We used text replays to label student behavior within a Cognitive Tutor for Algebra. 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 [15].

Within our text replays, we labeled whether a student was gaming the system. We chose gaming the system, because it is a behavior for which multiple classifiers have been developed and validated [6, 18, 20]. Within Cognitive Tutor Algebra, as with other Cognitive Tutors, gaming the system consists of systematic guessing and hint abuse (clicking through hints at high speed until the system gives the answer).

We obtained data from students' use of Cognitive Tutor Algebra from the Pittsburgh Science of Learning Center DataShop (https://learnlab.web.cmu.edu/datashop/). The DataShop is a public resource for the learning science community, giving free access to anonymized data sets of student use of learning software. We used the "Algebra I 2005-2006 (Hampton only)" data set, a data set consisting of 436,816 student actions (an action is defined as entering an answer or requesting help) made by 59 students over the course of an entire school year, in a high school in the suburbs of a city in the Northeastern USA. Data from 32 tutor lessons was obtained; however, 12 lessons were eliminated from consideration for one of two reasons: either having insufficient data to be able to conduct at least 498 observations (500 was the planned cut-off, but one lesson with 498 observations was included), or gaming the system never being coded in that lesson.

Across lessons, 18,737 text replays were labeled. These text replays were randomly selected; the chance of a specific sequence of actions being chosen was weighted, for each lesson, according to the total number of actions in that lesson. An average of 887 text replay observations were made in each lesson (SD=178; min=498; max=1075).

A single coder (the second author) labeled all 18,737 text replays. This coder was trained by the first author, who had previously used the technique in [7]. Training consisted of repeatedly coding the same clips, and comparing and discussing the results, until 100% agreement was obtained on two sets in a row. Text replays of gaming previously achieved reasonably high inter-reliability (Cohen's κ) of 0.58, in a study where coders did not engage in this training practice [7]. Training took approximately two hours.

The coder labeled the 18,737 text replays in 206 hours of work, a rate of about 39.6 seconds per observation. The 206 hours included time spent studying each tutor lesson's user interface. This was about twice as fast, label per label, as quantitative field observations of gaming conducted in prior research [7], and about 6 times as fast as observations of affect conducted using screen replays [11]. The observations were conducted across 4.5 months, at the coder's convenience, rather than during an entire school year during school hours. The time savings from conducting the observations when convenient for the coder – rather than when convenient for the school – was quite substantial. 4.5 months of work was able to generate data to build gaming classifiers for 20 tutor lessons (the classifiers will be discussed in the next section). By comparison, 3 years of work using quantitative field observations to label student data only produced enough data to develop classifiers of gaming for 4 tutor lessons [6]. Hence, text replays enabled us to collect data for 5 times as many classifiers in 12.5% as much total time – around 40 times faster. This result suggests that the text replay method is substantially more efficient than prior methods for generating labels of gaming the system, beyond just the time spent on each individual label.

5 Developing Classifiers of Gaming the System

Once the text replays were obtained, we developed two types of classifiers of gaming the system. First, we built classifiers of gaming using a Latent Response Model(LRM) [16]. In this approach, discussed in considerably more detail in [6], the model predicts whether each student action is gaming, then aggregates those predictions to predict each student's frequency of gaming (and trains on the overall proportion of time each student spent gaming, rather than training on labels of individual actions). Training detectors in this fashion enables us to study whether detectors developed using data from text replays are as accurate as detectors previously developed using data from quantitative field observations [6], using the same classification method.

Second, we took advantage of the fact that text replays label individual actions in the logs as gaming or not gaming, to use a second method. We built gaming detectors using J48 decision trees, the same technique used in Walonoski & Heffernan [20], using the Weka data mining package [21]. This method was slightly to moderately more accurate at classifying gaming the system than other classification algorithms within the Weka package. It is worth noting that these labels of individual actions cannot be considered perfectly accurate, since the observer labeled a clip as "gaming" if any of the actions in the clip involved gaming. Therefore, actions at the beginning or end of clips may not in all cases be instances of gaming. This suggests that, within this data set, a 100% perfect match between our classifier's labels of individual actions and those actions' labels (which will be measured with Kappa) is not necessary (or desirable). This limitation could be addressed in future work by having observers explicitly label which actions in a clip are gaming, but would have the cost of reducing the method's speed.

Within each approach, a separate classifier was trained for each lesson and tested in that lesson. Due to the large number of models fit, the training set was used as the test set in all cases, for tractability. As this testing method risks some over-fitting, pruning was used on the J48 trees, and model size was capped for the latent response models.

26 features were distilled for each action in the log files for use by the classification algorithms, including features involving details about the action (Was it correct or incorrect, or a help request? Was it the first attempt at the step?), assessments of student knowledge, information about the time the student took on this step (absolute, and relative to other students), and information about the student's previous interaction behaviors (How often has this student gotten this step wrong in the past? How much help has the student requested?). A full list of features can be found in [6].

6 Classifier Goodness

We assessed the two types of classifiers, for each lesson, using three metrics: A', correlation, and kappa. A' addresses a classifier's ability to assess which students gamed. A' is the probability that if the detector is comparing two students, one who gamed and one who never gamed, it will correctly identify which student is which. A' is equivalent to both the area under the ROC curve in signal detection theory, and to W, the Wilcoxon statistic [14]. A model with an A' of 0.5 performs at chance, and a model with an A' of

Table 1. The performance of each model on each metric, across models (standard errors given in parentheses). Both models are statistically better than chance on all metrics; boldface indicates a model is statistically significantly better than the other model, for that metric.

	A'	Correlation	Kappa
Latent Response Model	**0.96 (0.02)**	**0.90 (0.02)**	0.04 (0.01)
Decision Tree	0.69 (0.03)	0.44 (0.05)	**0.40 (0.05)**

1.0 performs perfectly. Correlation assesses whether the detector predicts the correct amount of gaming for each student. Kappa assesses whether the detector accurately identifies the correct actions as gaming. A Kappa of 0 indicates that the detector performs at chance, and a Kappa of 1 indicates that the detector performs perfectly. The full pattern of results is shown in Table 1.

For A', the Latent Response Models averaged an A' of 0.96 (SE=0.02) across the 20 lessons, meaning that they could distinguish a gaming student from a non-gaming student in the data 96% of the time. This A' is considerably higher than the A' of previous models of gaming the system developed using data from quantitative field observations (in that earlier work, models averaged an A' of 0.86 on the training set) [6]. The Decision Trees averaged an A' of 0.69 (SE=0.03). We can calculate the statistical significance of the difference of these models from chance (and from each other) by computing the standard error of each A' estimate [14], conducting Z tests, and then using Strube's Adjusted Z to aggregate across lessons in a fashion that controls for within-student non-independence [20, Appendix II in 6]. We find that the LRM A' is significantly higher than chance, $Z=26.56$, two tailed $p<0.0001$, that the Decision Tree A' is also significantly higher than chance, $Z=3.24$, two tailed $p<0.01$, and that the LRM A' is significantly higher than the Decision Tree A' across lessons, $Z=3.35$, two tailed $p<0.01$.

For correlation, the Latent Response Models averaged a correlation of 0.90 (SE=0.02) across the 20 lessons, between the predicted and actual frequencies of gaming, and the Decision Trees averaged a correlation of 0.44 (SE=0.05). Statistical significance can be computed by computing the significance of each correlation (or the significance of differences in correlation for correlated samples – [cf. 13]), converting those significance values to Z scores, and then aggregating across Z scores using Strube's Adjusted Z. We find that the LRM correlation is significantly higher than chance, $Z=10.04$, two tailed $p<0.0001$, that the Decision Tree correlation is also significantly higher than chance, $Z=3.46$,two-tailed $p<0.0001$, and that the LRM correlation is significantly higher than the Decision Tree correlation across lessons, $Z=5.68$, two tailed $p<0.0001$.

For kappa, the Latent Response Models averaged a kappa of 0.04 (SE=0.01) across the 20 lessons in predicting the labels of whether individual actions involved gaming, and the Decision Trees averaged a kappa of 0.40 (SE=0.05). The statistical test most closely corresponding to kappa is chi-squared; however, this data violates the independence assumptions of chi-squared. Thus, we instead do t-tests on the kappa values for each lesson (either one-sample t-tests or paired t-tests), convert the significance values to Z scores, and then aggregating across Z scores using Strube's Adjusted Z. This method treats the kappa values as point-estimates rather than as having a standard error,

effectively giving us a sample size of 20 (the number of lessons) rather than the full sample size – in other words, this test is overly conservative, and should err in the direction of non-significance. Despite the poor kappa for LRM, it is significantly higher than chance, $t(19)=3.77$, two tailed $p<0.01$. The Decision Tree kappa is also significantly higher than chance, $t(19)=7.81$,two-tailed $p<0.0001$. The Decision Tree kappa is significantly higher than the LRM kappa across lessons, $t(19)=7.45$, two tailed $p<0.0001$.

The poor performance of the LRM-based classifiers on kappa is surprising, given the high performance on the other metrics, and the effectiveness of this technique at assigning interventions to students which improve learning [4]. One potential explanation is that the model's use of historical features leads it to identify gaming in the correct skills, but on transactions after the transaction where gaming actually occurred (i.e. the next time the skill is encountered), a pattern observed in previous classifiers of gaming trained in this fashion [6]. Indeed, we find that the LRM-based classifier is significantly more likely to detect gaming on skills the student has previously gamed (i.e. detecting gaming after the fact), than on other skills, $t(58) = -4.77$, two-tailed $p<0.001$, for a paired t-test. Classifiers with this characteristic will be useful for some types of interventions, such as giving supplementary exercises on material a student bypassed by gaming [4] or giving metacognitive interventions to gaming students between problems [3], but will be less appropriate for more immediate types of interventions. Interventions which depend on occurring exactly after a student games should probably be assigned by detectors trained on individual actions rather than overall proportions of behavior. On the other hand, the LRM-based classifier's higher ability to identify gaming students is also relevant, since it increases the probability of assigning interventions to the students who most need them. Similarly, the degree to which this limitation will affect this classifier's usefulness for data mining analyses of motivation or learning depends on the analysis. Analyses at broader grain-sizes which look at the overall incidence of gaming [e.g. 8, 12] may benefit from using an LRM-based classifier, whereas analyses that depend on accurate assessments of whether each action involves gaming may benefit more from using a Decision Tree.

One potential solution that could avoid the limitations of the classifiers presented here would be to develop a hierarchical classifier (such as LRM) which optimizes on measures at both grain-sizes during training (e.g. using all three measures presented here). Such a classifier could potentially succeed both at detecting when an individual student is gaming, and distinguishing with maximum accuracy which students game and how much they game. However, this classifier would require the type of labels at multiple grain-sizes that Text Replays can provide.

7 Discussion and Conclusions

In this paper, we have presented text replays, a method for labeling student's behavior within log files. When all factors are taken into account, including the time needed to make individual labels, and the logistics involved in conducting studies in schools, text replays appear to speed the process of data collection by around 40 times compared to quantitative field observations, and at least 6 times compared to screen replays.

Where this technique applies, therefore, it may make classifier development accessible to a much larger pool of researchers and developers. However, this technique may not be applicable for coding every type of construct that can be coded with quantitative field observations or screen replays. Those techniques have been used to code students' affect and motivation [cf. 11, 17]; it is likely that some affective and motivational constructs will be difficult to code using text replays. Establishing which categories of behavior can accurately be labeled in text replays will be an important area for future work.

Unlike quantitative field observations, text replays make it easy to label individual actions as gaming or not gaming, though these labels are not 100% accurate, as clips often do not coincide exactly with the beginning and end of a gaming episode. Nonetheless, these action-by-action labels make it much more feasible to use off-the-shelf classifiers such as J48 Decision Trees to train gaming detectors which are reasonably successful at identifying exactly when students game, which students are gaming, and how much each student is gaming. Being able to use off-the-shelf classification algorithms available in Weka should make it more feasible for researchers and developers without machine learning experience to develop classifiers which are useful for learning analyses and to drive interventions by learning software.

An alternate modeling framework, Latent Response Models, was trained on each student's frequency of gaming. LRMs, as implemented here, were much more successful at identifying which students gamed, and how much each student gamed, but much less successful at identifying exactly when students game. Interestingly, the LRMs developed using text replay data appeared to be more accurate than prior LRMs developed using quantitative field observations. One explanation is that the text replay method both enables more precise assessments of students' gaming frequency (via making it easy to collect more labels), and allows a coder to take extra time on more ambiguous cases (while working quickly on clear cases).

The pattern of successes of the classifiers trained using different data suggests that it may be necessary to develop an algorithm that uses both overall gaming frequency data and action-by-action labels of gaming during the training processes in order to optimally capture both which students game, and when they game.

One curious aspect of the models developed here is that they do not replicate the split between types of gaming found in [6]. In that work, LRMs trained to classify gaming students systematically only captured a proportion of students – the students who gamed and had poorer learning. (A separate classifier was trained to detect students who gamed but did not have poorer learning). There was no evidence for such a split in this study, where LRM classifiers had average A' values of 0.96. It is not yet clear whether the split in gaming behavior in [6] was idiosyncratic to the learning system studied, or whether this effect is dependent on differences in the labeling method in some subtle fashion.

In conclusion, text replays appear to considerably speed the process of collecting the data necessary to develop effective classifiers of student behavior. Making it easier to develop classifiers of student behavior will expand the benefits of this technique to a wider pool

of researchers, facilitating the development of more adaptive educational software and the use of data mining techniques to conduct complex analyses on student learning.

Acknowledgements

We would like to thank Mercedes Rodrigo and Albert Corbett for helpful suggestions and discussions. This work was funded by the Pittsburgh Science of Learning Center, National Science Foundation award SBE-0354420.

References

[1] Aleven, V., McLaren, B.M., Roll, I., and Koedinger, K.R. Toward tutoring help seeking: Applying cognitive modeling to meta-cognitive skills. *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS 2004)*, 227-239.

[2] Amershi, S., Conati, C. Automatic Recognition of Learner Groups in Exploratory Learning Environments. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*, 463-472.

[3] Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S., and Woolf. B.P. Repairing Disengagement with Non-Invasive Interventions. *Proceedings of the 13h International Conference on Artificial Intelligence in Education (AIED-2007)*, 195-202.

[4] 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 (ITS 2006)*, 392-401.

[5] 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: Computer-Human Interaction*, 383-390.

[6] Baker, R.S.J.d., Corbett, A.T., Roll, I., Koedinger, K.R. (to appear) *Developing a Generalizable Detector of When Students Game the System* To appear in User Modeling and User-Adapted Interaction. Online at http://www.cs.cmu.edu/~rsbaker/USER475.pdf

[7] Baker, R.S.J.d., Corbett, A.T., 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*, 2006, 29-36.

[8] Baker, R., Walonoski, J., Heffernan, N., Roll, I., Corbett, A., Koedinger, K. *Why Students Engage in "Gaming the System" Behavior in Interactive Learning Environments*. Journal of Interactive Learning Research, 2008, 19 (2), 185-224.

[9] Beck, J. Engagement tracing: using response times to model student disengagement. *Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 2005)*, Amsterdam, pp. 88-95.

[10] Cocea, M., Weibelzahl, S. Eliciting Motivation Knowledge from Log Files Towards Motivation Diagnosis for Adaptive Systems. *Proceedings of 11th International Conference on User Modeling, UM2007,* 197-296.

[11] de Vicente, A. and Pain, H. Informing the detection of the students' motivational state: an empirical study. *Proceedings of the Sixth International Conference on Intelligent Tutoring Systems*, 2002, 933-943.

[12] Feng, M., Heffernan, N.T., Koedinger, K.R. Looking for Sources of Error in Predicting Student's Knowledge. *Educational Data Mining: Papers from the 2005 AAAI Workshop,* 54-61.

[13] Ferguson, G.A. *Statistical Analysis in Psychology and Education,* 1971. New York: McGraw-Hill.

[14] Hanley, J.A., McNeil, B.J. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology,* 1982, 143, 29-36.

[15] Koedinger, K. R., & Corbett, A. T. Cognitive tutors: Technology bringing learning sciences to the classroom. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences*, 2006, pp. 61-77. New York, NY: Cambridge University Press.

[16] Maris, E. Psychometric Latent Response Models. *Psychometrika*, 1995, 60 (4), 523-547.

[17] Rodrigo, M.M.T., Baker, R.S.J.d., Lagud, M.C.V., Lim, S.A.L., Macapanpan, A.F., Pascua, S.A.M.S., Santillano, J.Q., Sevilla, L.R.S., Sugay, J.O., Tep, S., Viehland, N.J.B. Affect and Usage Choices in Simulation Problem Solving Environments. *Proceedings of Artificial Intelligence in Education 2007*, 145-152.

[18] Roll, R., Baker, R., Aleven, V., McLaren, B., Koedinger, K. Modeling Students' Metacognitive Errors in Two Intelligent Tutoring Systems. *Proceedings of User Modeling 2005*, 367-376.

[19] Strube, M.J. Combining and comparing significance levels from nonindependent hypothesis tests. *Psychological Bulletin*, 1985, 97, 334-341.

[20] Walonoski, J.A., Heffernan, N.T. Detection and Analysis of Off-Task Gaming Behavior in Intelligent Tutoring Systems. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* 2006, 382-391.

[21] Witten, I.H., Frank, E. *Data Mining: Practical machine learning tools and techniques,* 2005. San Francisco: Morgan Kaufmann.

Adaptive Test Design with a Naive Bayes Framework

Michel C. Desmarais, Alejandro Villarreal, and Michel Gagnon
Computer and Software Engineering Department
Polytechnique Montréal
{michel.desmarais, alejandro.villarreal, michel.gagnon}@polymtl.ca

Abstract. Bayesian graphical models are commonly used to build student models from data. A number of standard algorithms are available to train Bayesian models from student skills assessment data. These models can assess student knowledge and skills from a few observations. They are useful for Computer Adaptive Testing (CAT), for example, where the test items can be administered in order to maximize the information they will provide. In practice, such data often contains missing values and, under some circumstances, missing values far outnumber observed values. However, when collecting data from test results, one can often choose which values will be present or missing by a consequent test design. We study how to optimize the choice of test items for collecting the data that will be used for training a Bayesian CAT model, such as to maximize the predictive performance of the model. We explore the use of a simple heuristic for test item choice based on the level of uncertainty. The uncertainty of an item is derived from its initial probability of success and, thus, from its difficulty. The results show that this choice does affect model performance and that the heuristic can lead to better performance. Although the study's results are more exploratory than conclusive, they suggest interesting research avenues.

1 Introduction

Applications such as study guides [6] and adaptive tutoring [7, 10] must rely on a fine grained student model to tailor their interaction with the user [2]. Bayesian models data can be used for this purpose and they can be trained with skills assessment data to overcome a knowledge engineering effort [12, 13, 9, 4]. In many contexts, such as Computer Adaptive Testing (CAT), this data will contain missing values. We investigate the problem of choosing which items will be administered in each test with the aim of optimizing the choice of missing values distribution.

In CAT, this problem is often referred to as sampling design or test design/construction. See for example [1, 8, 11]. The goal of student modeling in CAT is to build a statistical model of an examinee's chances of success or failure to a test based on previous observations of correct or incorrect answers to a subset of test items. For a number of

practical reasons, the pool of test items often needs to be quite large, such as a few hundreds of items. However, for model training, it is impractical to administer a test of hundreds of questions to examinees in order to gather the necessary data, as such test could last tens of hours. We are thus forced to administer a subset of these test items to each examinee and, hence, we run into the problem of choosing how to distribute the test items in order to maximise the information we get for model building.

The work on test design is generally conducted within the Item Response Theory framework. We investigate this issue in the context of the Bayesian framework.

The next section describes the Bayesian model used for this study. Then, a few heuristics are proposed to design the missing values scheme that we hope will bring the most relevant information for model building given a fixed number of observations. Experimental studies of the performance of these heuristics are later reported and further studies are proposed.

2 The POKS Naive Bayes Framework

To investigate the issue of test design within the Bayesian framework, we use the POKS framework [4] (Partial Order Knowledge Structures). This model is fully automated and it was shown to perform well compared to other Bayesian student frameworks that involve a knowledge engineering effort, while providing much finer grained assessment than IRT [3].

This model is based on the Naive Bayes conditional independence assumption.

The POKS model aims to predict the outcome of each individual item based on the observed items. For each item, the model determines which set of evidence items, $\mathbf{X_e}$, are considered relevant to determine success to item X_c:

$$P(X_c|\mathbf{X_e}) = P(X_c|X_1, X_2, \ldots, X_k)$$

where $\{X_1, X_2, \ldots, X_k\}$ is the set of evidence items considered as relevant. Given the conditional independence assumption mentioned, that equation becomes:

$$P(X_c|\mathbf{X_e}) = \frac{P(X_c)}{P(X_1, X_2, \ldots, X_k)} \prod_i^k P(X_i|X_c)$$

To determine which items are included in $\mathbf{X_e}$ three statistical tests are applied for each possible pairs of nodes, (X_c, X_i). Two of these tests aim to verify the strength of the conditional probabilities $P(X_e|X_c) \geq p_c$ and $P(\overline{X_c}|\overline{X_e}) \geq p_c$, and another one to verify the interaction between X_c and X_e. See [4] for details.

This approach has been shown to perform at least as well as the IRT framework for predicting global test mastery [5] and also provides a fine grained student model.

3 Heuristics for Effective Partial Data Sampling Scheme

As explained above, the context of adaptive testing is a typical case where we have the opportunity to decide upon a specific scheme of missing values for each item. We can decide which subset of questions we wish to administer to each examinee, leaving unanswered items as missing values. For a pool of hundreds of question items, which is a size often found in CAT, each examinee can be administered only 50 question items, leaving many more missing values than answered items.

Given that we have the choice of how many observations will be assigned to each item, we need to determine which item are most critical and, potentially, ought to be allotted more observations.

Without knowing in advance the topology of the POKS networks, or the topology of a Bayes Network over the items, we have to revert to simpler means of choosing critical items. Potential candidates are the highly uncertain items that have an initial probability of 0.5. They correspond to items of average difficulty. To the opposite spectrum, we can also favor choosing low uncertainty items that have a high or a low initial probability. They correspond to the most difficult and the easiest items. In this study, we do not discriminate between easy and difficult items, as, in terms of uncertainty, or entropy, they are equivalent. Of course, further investigations could explore such distinction.

Initial probabilities can be obtained from relatively small samples, in contrast to joint probabilities, so it is reasonable to assume that a pilot sample can provide these with sufficient reliability to guide further sampling.

We have conducted a simulation study of such a sampling scheme. The details of the experimental conditions and the results are described below.

4 Experimental Design

We define three sampling schemes to determine missing values in order to investigate their respective effects over the predictive accuracy of POKS Bayesian models :

1. *Uniform*: Uniform random samples of missing values.
2. *Most uncertain*: Higher sampling rate of missing values for uncertain items (favoring the choice of average difficulty items).
3. *Least uncertain*: Lower sampling rate of missing values for uncertain items (favoring the choice of difficult and easy items alike).

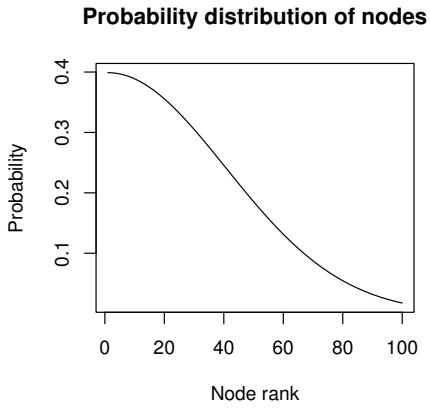


Figure 1: Sampling probability distribution of items used for the *most uncertain* and *least uncertain* sampling schemes.

Uncertain items are the items that, obviously, are closest to an initial probability of 0.5.

For the *most uncertain* and *least uncertain* conditions, the probability of sampling is based on the $x = [0, 2.5]$ segment of a normal (Gaussian) distribution as reported in Figure 1. The probability of an item being sampled will therefore vary from 0.40 to 0.0175 as a function of its rank, from the most to the least uncertain item on that scale. Items are first ranked according to their uncertainty and they are attributed a probability of being sampled following this distribution. The distributions are the same for both conditions (2) and (3), but the ranking is reversed between the two of them. For the *uniform* condition (1), all items have equal probability of being sampled.

Ten samples are created according to the three sampling schemes above. They are used to validate the effect of the sampling scheme by performing CAT simulations and measuring the predictive power of the models based on different sampling schemes.

4.1 Simulation process

The experiment consists in simulating the question answering process with the real subjects. An item is chosen and the outcome of the answer, success or failure, is fed to the inference engine (POKS). An updated probability of success is computed given this new evidence. All items for which the probability is above 0.5 are considered mastered and all others are considered non-mastered. We then compare the results with the real answers to obtain a measure of how accurate the predictions are. The process is repeated from 0 item administered until all items are "observed". Observed items are bound to their true value, such that after all items are administered, the score always converges to 100%.

The simulations replicate a context of computer adaptive testing (CAT) where the system chooses the question items in order to optimize skills assessment. The choice of item relies on a measure of the most informative question that gets administered to the examinee. This can be achieved in a number of ways and the results are often relatively close. We

use a heuristic that our exploratory results has shown to approach the performance of the information gain approach (see [5]), but which is computationally much faster. It consists in choosing the item i that has a high entropy and is highly connected to other nodes:

$$\max_i \; E(i)\frac{\log(links(i+1)+E(0.5))}{E(0.5)}$$

where $E(i)$ is the entropy of item i and $links(i)$ is the number of incoming and outgoing links. The use of the maximal entropy of a item, $E(0.5)$, in the above equation is a normalizing factor that ensures the weights between the number of links and the entropy are similar.

Once the outcome of the answer to a question item is obtained, the probability of success to each other questions is then recalculated according to the POKS framework described above.

Simulations consist in ten-fold cross-evaluation runs. Each run consists of a different random sampling for test design (the choice of items according to the three schemes described in section 4) and a different random sampling of the examinee used for training and testing. We report the average results of the 10 simulations for each experimental condition.

4.2 Data sets

Four data sets are used for the simulations. They are based on real data from tests in four different domains :

Table 1 reports general statistics on these data sets as well as the sizes of the training and testing samples used for the simulations.

Table 1: Data sets

	Data Set	Set size		Number of items	Average success rate
		Training	Testing		
1	College math	375	56	60	61%
2	UNIX	30	18	34	53%
3	Arithmetic	100	49	20	61%
4	French	25	17	160	58%

The proportion of missing values inserted in the training set is half of the data. The testing data sets contain no missing values.

1. *College math*: a 60 question items test covering different topics in mathematics, from general high school math, to college level geometry, linear algebra, and calculus. The test was administered to 426 candidates newly admitted at an engineering school.

2. *UNIX*: a 34 question items test covering knowledge of the UNIX shell commands, from the basic "change directory" (`cd`) to advanced data manipulation commands with `awk`. The test was administered to 48 individuals with a wide variety of knowledge about UNIX.

3. *Arithmetic*: a 20 questions test on basic fraction arithmetic. The test was administered to 149 pupils in grade 10 to 12. More details can be found in [13].

4. *French*: a 160 general French grammar, reading, and comprehension test administered to 42 adults.

5 Results

The results of the simulation experiments are reported in Figures 2(a) to 2(d). The Y axis represents the proportion of correct predictions while the X axis reports the number of items administered. As mentioned, administered items are considered correctly classified, and thus, after all items are administered, the score reaches 100%. Given that the items are initialized to their unconditional probabilities, the prediction score generally starts above 70%, which indicates that more than two thirds of items are already correctly classified initially. A 90% confidence interval over the 10 simulations is reported around each data point.

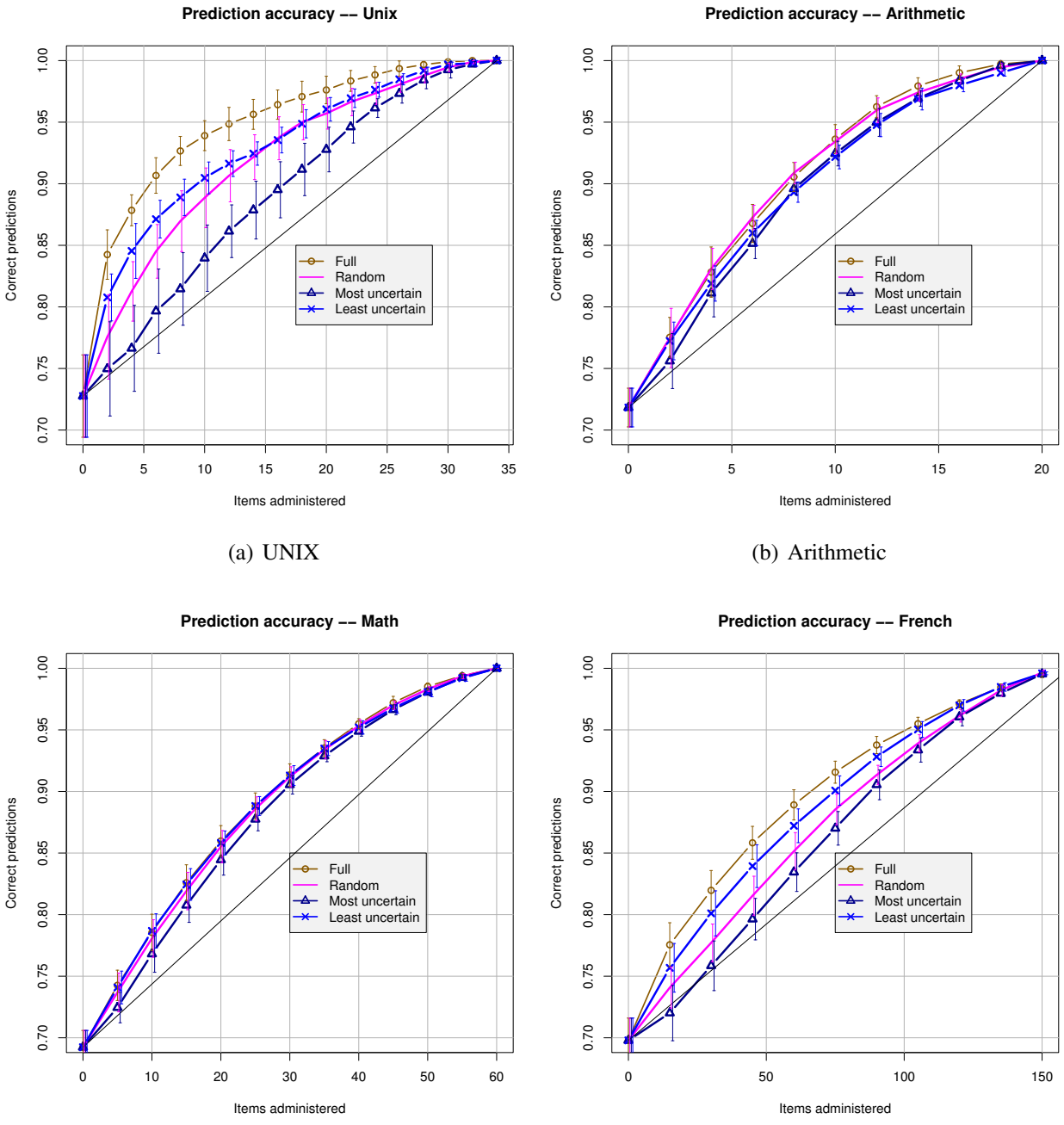
Each figure contains four curves. For comparison purpose, we report the *Full* condition which corresponds to the results for the full data set, without any missing values. The other three conditions are described in section 4.

The results show non significant differences for the *Arithmetic* and *College* math data sets. However, more significant differences are observed for the other two data sets (*UNIX* and *French*), and they follow a regular pattern: The *least uncertain* condition systematically outperforms the *most uncertain* condition, which, in turn, performs systematically worst than the *uniform condition*. As expected, the *full* data set is systematically better than, or equal to, the data sets that contains missing values.

6 Discussion

These results suggest that higher sampling rates for the *least uncertain* items generally bring a higher predictive performance than for the *most uncertain* or the *uniform* choice, although this gain is not systematic.

These results remain exploratory and a number of questions are left open. For one, how should we explain the patterns of differences found between the *least uncertain*, and the *most uncertain*? We initially hypothesized that the most uncertain items are the ones that would benefit the most from a higher sampling frequency. These items are generally the ones that bring the most information, and it seems reasonable to gather more data for them to correctly establish their relations to other items. However and contrary to these



(a) UNIX

(b) Arithmetic

(c) College math

(d) French

Figure 2: Results of the four data test. 90% confidence intervals are displayed around each data point.

expectations, higher sampling of uncertain items yields the models with the poorest performance. One plausible explanation is that the estimation of the model's conditional probabilities is more subject to noise and to miscalibration for probabilities closer to 0 or 1 than for mid-range probabilities. As a consequence, a higher sampling rate for these items is required. This hypothesis also explains why we observe a large difference between *least uncertain* and *most uncertain* for small data sets (UNIX and French) than for the larger ones (College math and French): larger data sets are not as subject to sampling noise as smaller ones are.

Another open question is whether these results apply to other domains and to other Bayesian frameworks for student modeling. For example, would the results be the same if we used a more general Bayesian Network approach that captures independence relations, such as in [13]?

A potentially interesting question to investigate is the hypothesis that the items that would be most informative are the ones that are central and highly connected in a Bayesian Network, that is, the nodes that are likely to influence the greatest number of nodes in the network. These nodes could benefit from a higher sampling rate. Moreover, given an initial topology of a Bayesian Network, we could guide the sampling beyond individual nodes, to pairs or to n-tuples of nodes that are deemed more critical.

However, the topology of a Bayesian network might not be reliably established with small sample sizes, in contrast to the heuristic that we used which is based on estimating the individual items non conditional probabilities: they require relatively small sample size. It is feasible to design the tests with an initial sample of a few tens of data records, and then collect a larger sample for estimating the conditional, joint probabilities. Whether this can be effectively done for a Bayesian Network remains open.

Further analysis and investigations are obviously required to bring some understanding to these results. Nevertheless, this investigation shows that we can influence the predictive performance of a Naive Bayes framework with partial data when we have the opportunity to select the missing values. It opens interesting questions and can prove valuable in some contexts of application.

References

[1] Berger, M. P. F. A general approach to algorithmic design of fixed-form tests, adaptive tests, and testlets. *Applied Psychological Measurement 18*, 2 (1994), 141–153.

[2] Bra, P. D., Brusilovsky, P., and Houben, G.-J. Adaptive hypermedia: from systems to framework. *ACM Comput. Surv. 31*, 4es (1999), 12.

[3] Desmarais, M. C., Gagnon, M., and Meshkinfam, P. Item to item student models. In *AAAI'2006 Workshop 06, Data mining in education* (Sept. Boston MA 2006), p. 10.

[4] Desmarais, M. C., Maluf, A., and Liu, J. User-expertise modeling with empirically derived probabilistic implication networks. *User Modeling and User-Adapted Interaction 5*, 3-4 (1996), 283–315.

[5] Desmarais, M. C., and Pu, X. A bayesian inference adaptive testing framework and its comparison with Item Response Theory. *International Journal of Artificial Intelligence in Education 15* (2005), 291–323.

[6] Falmagne, J.-C., Cosyn, E., Doignon, J.-P., and Thiéry, N. The assessment of knowledge, in theory and in practice. In *ICFCA* (2006), R. Missaoui and J. Schmid, Eds., vol. 3874 of *Lecture Notes in Computer Science*, Springer, pp. 61–79.

[7] Heller, J., Steiner, C., Hockemeyer, C., and Albert, D. Competence–based knowledge structures for personalised learning. *International Journal on E–Learning 5*, 1 (2006), 75–88.

[8] Henson, R., and Douglas, J. Test construction for cognitive diagnosis. *Applied Psychological Measurement 29*, 4 (2005), 262–277.

[9] Millán, E., de-la Cruz, J.-L. P., and Suárez, E. Adaptive Bayesian networks for multilevel student modelling. In *ITS'00: Proceedings of the 5th International Conference on Intelligent Tutoring Systems* (2000), Springer-Verlag, pp. 534–543.

[10] Millán, E., Garcia-Herve, E., Rueda, A., and de-la Cruz, J. P. Adaptation and generation in a web-based tutor for linear programming. *Lecture Notes in Computer Science 2722* (January 2003), 124–127.

[11] Mislevy, R. J., Beaton, A. E., Kaplan, B., and Sheehan, K. M. Estimating population characteristics from sparse matrix samples of item responses. *Journal of Educational Measurement 29*, 2 (The National Assessment of Educational Progress (Summer, 1992) 1992), 133–161.

[12] Pardos, Z. A., Feng, M., Heffernan, N. T., Lindquistheffernan, C., and Ruiz, C. Analyzing fine-grained skill models using bayesian and mixed effects methods. In *Workshop of Educational Data Mining* (Los Angeles 2007).

[13] Vomlel, J. Bayesian networks in educational testing. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems 12*, Supplementary Issue 1 (2004), 83–100.

Interestingness Measures for Association Rules in Educational Data

Agathe Merceron[1] and Kalina Yacef[2]
merceron@tfh-berlin.de, kalina@it.usyd..edu.au,
[1]Computer Science and Media Department,
University of Applied Sciences TFH-Berlin, Germany
[2]School of Information Technologies, University of Sydney, Australia

Abstract. Educational data differs from traditional knowledge discovery domains in several ways. One of them is the fact that it is difficult, or even impossible, to compare different methods or measures a posteriori and deduce which the best is. It is therefore essential to use techniques and measurements that are fairly intuitive and easy to interpret. Extracting the most interesting association rules can be quite tricky. One of the difficulties is that many measures of interestingness do not work effectively for all datasets and are hard to understand intuitively by the teachers. We argue in this paper that cosine and added value (or equivalently lift) are well suited to educational data, and that teachers can interpret their results easily. We argue that interestingness should be checked with cosine first, and then with lift if cosine rates the rule as non-interesting. If both measures disagree, teachers should use the intuition behind the measures to decide whether or not to dismiss the association rule. We provide a case study with data from a LMS.

1 Introduction

Educational data mining differs from knowledge discovery in other domains in several ways. One of them is the fact that it is difficult, or even impossible, to compare different methods or measures a posteriori and decide which is the best. Take the example of building a system to transform hand-written documents into printed documents. This system has to discover the printed letters behind the hand-written ones. It is possible to try several sets of measures or parameters and experiment what works best. Such an experimentation phase is difficult in the educational field because the data is very dynamic,can vary a lot between samples and teachers just cannot afford the time and access to the expertise to do these tests on each sample, especially in real time. Therefore, as argued in [1], one should care about the intuition of the measures, parameters or methods used in educational data mining.

Another difference is the size of the data: while tremendous amounts of data are collected about students' work, the size of the data on one sample is usually small. Typically in a classroom there are at best a few hundreds students enrolled. Students may not all do the same exercises or activities. Collecting several years of data is certainly an option but there are instances where one wants to analyse the data as early as the first year. Besides, there are often changes between offerings of a course that have an impact on the common attributes of the data (for example not exactly the same topics/exercises/resources are offered from one year to the next). Therefore one should also be careful to avoid measures, parameters and methods where sample size has a predominant effect on the result.

Association rules are increasingly used in educational data mining [8, 9, 12]. However, measuring the interestingness of a rule can be problematic, as explained in [2]. Two measures, support and confidence, are commonly used to extract association rules. However it is well known that even rules with a strong support and confidence may in fact be uninteresting. This is why, once the association rule $X \rightarrow Y$ has been extracted, it is wise to double check how much X and Y are related. About 20 measures have been proposed in the literature to do so. Unfortunately, no measure is better than all the others in all situations, though measures tend to agree when support is high [11].

In this paper, we revisit measures of interestingness for association rules and argue that two of them, cosine and added value, are particularly suited for educational data because their meaning is fairly intuititive even to non data mining experts and one of them, cosine, does not depend on the data size. We present a case study using our usage data from the Learning Management System Moodle [10].

In the next section we present the underlying concepts: the association rules, the cosine, the added value and the lift, with typical values for each of these measures followed by a discussion. We then present a case study, corresponding to a common situation of using a LMS to put additional learning material for students. Supporting teachers in this task is giving them some way to be informed of the use of this extra material by students, and of its impact on the marks. Association rules are used to refine the findings made while exploring data. Cosine and lift agree on most rules found, but not on all. As we will see, the intuition behind the two measures helps to understand this information.

2 Association Rules, cosine, Added Value and Lift

Association rules come from market basket analysis and capture information such as "if customers buy book X, they also buy book Y". This can be written as $X \rightarrow Y$. Two measures characterize an association rule: support and confidence.

2.1 Association rules

Let $I = \{I_1, I_2, ..., I_p\}$ be a set of p items and $T = \{t_1, t_2, ..., t_n\}$ be a set of n transactions, with each t_i being a subset of I.

An *association rule* is a rule of the form $X \rightarrow Y$, where X and Y are disjoint subsets of I having a support and a confidence above a minimum threshold.

Let us denote by $|X, Y|$ the number of transactions that contain both X and Y. The support of that rule is the proportion of transactions that contain both X and Y: $sup(X \rightarrow Y) = |X, Y| / n$. This is also called $P(X, Y)$, the probability that a transaction contains both X and Y. Note that the support is symmetric: $sup(X \rightarrow Y) = sup(Y \rightarrow X)$.

Let us denote by $|X|$ the number of transactions that contain X. The confidence of a rule $X \rightarrow Y$ is the proportion of transactions that contain Y among the transactions that contain X: $conf(X \rightarrow Y) = |X, Y| / |X|$. An equivalent definition is : $conf(X \rightarrow Y) = P(X,Y) / P(X)$, with $P(X) = |X| / n$. This is also written $P(Y/X)$, the probability that a transaction contains Y knowing that it contains X already. Note that confidence is not symmetric,

usually *conf(X→Y)* is different from *conf(Y→X)*, and gives its direction to an association rule.

2.2 Cosine

Consider two vectors x and y and the angle they form when they are placed so that their tails coincide. When this angle nears 0°, then cosine nears 1, i.e. the two vectors are very similar: all their coordinates are pairwise the same (or proportional). When this angle is 90°, the two vectors are perpendicular, the most dissimilar, and cosine is 0.

Let *x* and *y* be two vectors of length *n*: $x=(x_1,\dots,x_n),\ y=(y_1,\dots,y_n)$.

Then $cosine(x,y)=\frac{(x\cdot y)}{(\|x\|.\|y\|)}$, where . above indicates the vector dot product $x\cdot y=\sum_{k=1}^{n} x_k y_k$ and $\|x\|$ is the length of vector *x*, $\|x\|=\sqrt{\sum_{k=1}^{n} x_k^2}$.

Borrowing this idea, it is easy to associate two vectors x and y to the rule *X→Y*. Let us interpret x_k as being *1* if transaction t_k contains *X* and *0* otherwise, and similarly for y_k and *Y*. Then it is immediate that the equation for cosine can be rewritten as $cosine(x,y)=\frac{P(X,Y)}{\sqrt{(P(X).P(Y))}}=cosine(X\rightarrow Y)$, the usual form that is given for cosine of an association rule *X→Y*. The closer *cosine(X→Y)* is to *1*, the more transactions containing item *X* also contain item *Y*, and vice versa. On the contrary, the closer *cosine(X→Y)* is to *0*, the more transactions contain item *X* without containing item *Y*, and vice versa. Simplifying with *n* gives $cosine(X\rightarrow Y)=\frac{|X,Y|}{\sqrt{|X|.|Y|}}$. This equality shows that transactions not containing neither item *X* nor item *Y* have no influence on the result of *cosine(X→Y)*. This is known as the null-invariant property. Note also that cosine is a symmetric measure.

2.3 Added value and lift

The added value of the rule *X→Y* is denoted by $AV(X\rightarrow Y)$ and measures whether the proportion of transactions containing *Y* among the transactions containing *X* is greater than the proportion of transactions containing *Y* among all transactions. Then, only if the probability of finding item *Y* when item *X* has been found is greater than the probability of finding item *Y* at all can we say that *X* and *Y* are associated and that *X* implies *Y*.

$AV(X\rightarrow Y)=P(Y/X)-P(Y)=conf(X\rightarrow Y)-P(Y)$. A positive number indicates that *X* and *Y* are related, while a negative number means that the occurrence of *X* prevents *Y* from occurring. Added Value is closely related to another well-known measure of interest, the lift.

$$lift(X\rightarrow Y)=\frac{P(X,Y)}{P(X).P(Y)}=\frac{conf(X\rightarrow Y)}{P(Y)}.$$

Note that if *P(X, Y) = P(X) . P(Y)* the lift is *1*. In terms of probability, this means that the occurrence of *X* and the occurrence of *Y* in the same transaction are independent events, hence *X* and *Y* are not correlated. It is easy to show that the lift is *1* exactly when added

value is *0,* the lift is greater than 1 exactly when added value is positive and the lift is below *1* exactly when added value is negative. Further *AV(X→Y)* tends towards *1* when *lift(X→Y)* tends towards infinity, and *AV(X→Y)* tends towards *-1* when *lift(X→Y)* tends towards *0.*

Note that $lift(X \rightarrow Y) = \frac{|X, Y|.n}{|X|.|Y|}$ so the result is proportional to *n*, the total number of transactions. As opposed to cosine, lift does not hold the null-invariant property.

2.4 Typical values for cosine and lift

To fix ideas let us look at typical values for these measures Suppose that among *n* transactions, *m* contain either *X* or *Y* or both, with $m \leq n$, and that *n* - *m* transactions contain neither *X* nor *Y*.

First consider the case where all *m* transactions contain both *X* and *Y*. Then: *cosine(X→Y) = 1*. Conversely, it is easy to show that *cosine(X→Y) = 1* implies that all *m* transactions contain both *X* and *Y*. As for the lift, $lift(X \rightarrow Y) = (m.n)/(m.m) = n/m$. So if *m = n*, *lift(X→Y)*=1. If *m = ½ . n*, *lift(X→Y)*=2 and so on.

Consider now the case where 90% of the *m* transactions contain both *X* and *Y*, and 10% of the rest contain *X* but not *Y*. Then:

$$cosine(X \rightarrow Y) = \frac{\frac{90}{100}.m}{\sqrt{m.\frac{90.m}{100}}} = \frac{90}{100}.\sqrt{\frac{100}{90}} = 0.949$$

$$lift(X \rightarrow Y) = (\frac{90}{100}.m.n)/(m.\frac{90}{100}.m) = \frac{n}{m}$$

Now consider again the case where 90% of the *m* transactions contain both *X* and *Y*, but 5% of the rest contain *X* and not *Y*, and the other 5% contain *Y* and not *X*. In other words *X* and *Y* are evenly spread among the transactions containing either *X* or *Y* but not both Then:

$$cosine(X \rightarrow Y) = \frac{\frac{90}{100}.m}{\sqrt{\frac{95.m}{100}.\frac{95.m}{100}}} = \frac{90}{100}.\frac{100}{95} = 0.947$$

$$lift(X \rightarrow Y) = (\frac{90}{100}.m.n)/(\frac{95}{100}m.\frac{95}{100}.m) = 0.99\frac{n}{m}.$$

Table 1 summarizes further results. Lines should be read as follows: (*a,b,c*) means that *a %* of the *m* transactions contain both *X* and *Y, b%* contain *X* and *c%* contain *Y*. Therefore (75, 100, 75) means that 75% of the *m* transactions contain both *X* and *Y* and that the remaining 25% contain *X* but not *Y* (*X* is present in 100% of the transactions and *Y* in 75% of them), while (75, 87.5, 87.5) means that *X* or *Y* are evenly spread among the 25% of the remaining transactions.

Discussion: In the case of strong symmetric association rules, which means that $|X|$, $|Y|$ and $|X, Y|$ are all big numbers close to *n*, cosine and lift do not rate rules the same way, as pointed out in [7]. In this case, cosine performs better than lift. Added value and

lift rely on probabilities, which make more sense when the number of observations is large. Further we see also that lift and added value, unlike cosine, depend on the number of transactions that contain neither X nor Y. In the educational field it is not clear that these null-transactions should play a role. We come to the same conclusion as [3]: double check the interestingness of association rules with cosine first, then with lift if cosine is not conclusive. Table 1 suggests that a value around or below 0.65 is rejected by cosine : as we can see 0.66 corresponds to the lowest threshold with 50% of common values (50, 75, 75). In case of contradictory results then decide using the information that these two measures represent.

Table 1. Typical values for cosine and lift, where the 3 figures of the first column show the percentage of transactions containing X and Y, X and Y

% transactions (X and Y, X, Y)	cosine(X→Y)	lift(X→Y)
(100, 100, 100)	1	n/m
(90, 100, 90)	0.949	n/m
(90, 95, 95)	0.947	0.997 . (n/m)
(75, 100, 75)	0.87	n/m
(75, 87.5, 87.5)	0.86	0.98 . (n/m)
(60, 100, 60)	0.77	n/m
(60, 80, 80)	0.75	0.94 . (n/m)
(50, 100, 50)	0.707	n/m
(50, 75, 75)	0.66	0.88 . (n/m)
(40, 100, 40)	0.63	n/m
(40, 70, 70)	0.57	0.82 . (n/m)
(30, 100, 30)	0.55	n/m
(30, 65, 65)	0.46	0.71 . (n/m)

3 Improving Teacher Support: Case Study

The present case study describes a standard use of a Learning Management System (LMS) for providing additional resources to students in a face-to-face teaching context. Teachers want to figure out whether students use these resources and possibly whether their use has any (positive) impact on marks.

The LMS Moodle [10] was used in the context of the course *Formal Basics of Computer Science* for first semester students enrolled in the degree "Computer Science and Media" at the University of Applied Sciences TFH Berlin during Winter Semester 2007/08. The cohort of 84 students enrolled in that course is divided into two groups. Students had a 3-hour weekly lecture. It includes formal teaching where concepts are explained, paper/pencil exercises to apply these concepts, and exercises discussed on the spot. To pass this course students take two exams. The first one takes place about 8 weeks after the beginning of the semester and the second one at the end of the semester. The present case study uses data gathered till the first exam.

Moodle is used for posting lecture slides and accessing the following extra resources:

- *Book*: a link to the homepage of the text book "Introduction to Automata Theory, Languages and Computation" used for this course [4]. From this homepage students could access a set of exercises with solutions.
- *DP*: extra reading "Design Patterns for finite automata" [6].
- *Jflap* [5], a software to practice automata construction.
- *Ex1, Ex2 ... Ex7* : a set of seven extra self-evaluation exercises. One exercise is published in Moodle each week right after the lecture. The last exercise *Ex7* was put 2 weeks before the exam.
- *TrEx01* and *TrEx02* : two sample exams, published 3 weeks before the exam.
- *TrEx01S* and *TrEx02S*, the solutions to the sample exams, published 10 days before the exam.

The use of Moodle, its additional resources and its self-evaluation exercises were not compulsory though strongly encouraged. Therefore for the teacher it is quite important to know: what do students do with those extra resources? What do they view? Is there any relationship between their use of these resources and their result in the exam? To answer these questions we have used solely the log data available in Moodle. Log data gives, for each resource and each student login, when the resource was accessed. It also gives, for each exercise and each student login, whether the exercise has been attempted, and whether the first trial was a success or not.

3.1 Exploring Data

From the 84 students enrolled in the course, 81 were enrolled in Moodle. The case study considers only those 81 students. From them, 52 passed the exam, 8 failed and 21 did not come. From the 60 who took the exam, statistics on their marks is given in the first line "General" of Table 4.

Did students do the exercises? Table 2 summarizes the figures. Lines should be read as follows. For example line 2 means that 46 students did not attempt exercise 1, 21 students gave a correct answer on their first trial and 14 gave a wrong answer on their first trial. One notices that as time goes there is always less students attempting exercises.

Table 2. Exploring exercises among all students.

Exercise	**Ex1**	**Ex2**	**Ex3**	**Ex4**	**Ex5**	**Ex6**	**Ex7**
No attempt	46	53	63	65	70	70	71
Success	21	19	11	8	6	9	8
Fail	14	9	7	8	5	2	2

Table 3. Viewing resources.

TrEx01	TrEx01S	TrEx02	TrEx02S	JFlap	DP	Book	AtLeast1Ex
59	52	53	47	39	36	23	38

Did they access other resources? Table 3 summarizes the figures. The first column says that 59 students have viewed the first sample exam, the second column says that 52

students have viewed the solution of the first sample exam, and so on. One extra column has been added. *AtLeast1Ex* says that 38 students have attempted at least 1 exercise.

What are the results in the exam for each group of Table 3? Table 4 summarizes the results. Two extra lines have been added. *NoEx* shows the results for students who have never attempted any exercise. *AtLeast1Ex* shows the results for the students who have attempted at least 1 exercise. Table 3 and Table 4 suggest that the standard preparation for the exam is to look at sample exams and/or their solutions. Students who invest some more time with extra material tend to have better marks. The biggest positive impact on the marks is given by *DP*.

Table 4. Viewing resources and marks in the exam.

Resource	minimum	maximum	average	s.deviation
General	11	50	36.45	10.85
TrEx01	14	50	36.86	10.57
TrEx01S	14	50	36.61	10.71
TrEx02	14	50	37.12	10.59
TrEx02S	14	50	36.73	10.79
JFlap	14	50	37.90	10.52
DP	14	50	44.08	8.83
Book	14	50	40.22	9.37
NoEx	11	50	33	10.91
AtLeast1Ex	14	50	39.09	10.18

Table 3 and 4 confirm the expected outcome. Table 4 also shows something that was not known before: students tend to access a sample exam more that its solution.

This first exploration gives also directions for more investigation: If students attempt exercise 2, do they also attempt exercise 1? If they look at the solution of a sample exam, do they also look at the sample exam itself? This kind of questions can be investigated with association rules.

3.2 Association Rules

We begin with association rules tackling sample exams. The following rules again confirm the expected finding. If students look at the solution of a sample exam, they look also at the sample exam itself. Further, if they view the second exam, then they also view the first one. The other way round does also hold, but with a slightly lower confidence.

Table 5. Association rules for sample exams

rule	sup.	conf.	cos.	lift
TrEx01S → TrEx01	0.59	0.92	0.87	1.27
TrEx01 → TrEx01S	0.59	0.81	0.87	1.27
TrEx02S → TrEx02	0.56	0.96	0.90	1.46

TrEx02 → TrEx02S	0.56	0.85	0.90	1.46
TrEx01S → TrEx01	*0.72*	*0.96*	*0.90*	*1.11*
TrEx01 → TrEx01S	*0.72*	*0.84*	*0.90*	*1.11*
TrEx02 → TrEx01	0.64	0.98	0.93	1.35
TrEx01 → TrEx02	0.64	0.88	0.93	1.35

Results are similar when rules are mined restricting the population to the students who came to the exam as shown for the first sample exam in the lines in italic of Table 5. Notice however that the lift diminishes as the rules become stronger [7].

Table 2 gives a direction for further rules to investigate: Is there any association between attempting exercise i and exercise j? One expects that many students enthusiastically have begun with exercise 1 at the beginning of the semester and then slowly have stopped doing them, till exercise 4 where a bunch of students just keep doing them. The rules we have obtained confirm this interpretation.

We have mined these rules restricting the data to students who have attempted at least 1 exercise, which means 38 transactions. Rules with a high confidence relate attempting exercise 2 and exercise 3, exercises 4 to 7, as well as exercise 1, and not attempting exercises 2 to 3, or not attempting exercises 4 to 7. Table 6 presents a sample of the extracted associations. Note that *!Ex2* means that exercise 2 has not been attempted. So the first line says if students don't attempt exercise 2, then they don't attempt exercise 3.

Table 6. Association rules for attempted exercises

rule	sup.	conf.	cos.	lift
!Ex2 → !Ex3	0.26	1	0.71	1.9
Ex3 → Ex2	0.47	1	0.80	1.36
!Ex4, !Ex5 → !Ex6	0.56	1	0.93	1.41
!Ex6, !Ex7 → !Ex5	0.63	0.96	0.92	1.35
Ex1, !Ex4 → !Ex5, !Ex6	0.55	1	0.90	1.46
Ex1, Ex7 → Ex5, Ex6	0.21	1	0.89	3.8
!Ex5, !Ex6 → Ex1	0.63	1	0.80	1.0

For all these rules, except the last one, cosine and lift rate associations the same way. The drop between attempting exercises 1 to 3 and attempting the others has led us to investigate the marks of this population. Surprisingly, their average mark is smaller than for all students who have attempted at least 1 exercise.

Table 7. Attempting exercises 4 to 7 and marks in the exam.

Resource	minimum	maximum	average	s.deviation
Ex 4 to 7	16	50	38.76	10.35

As for the other resources, were they consulted by the same students? We have looked at associations between *DP*, *Jflap*, *Book* and *AtLeast1Ex* considering the full population and

show two rules found in Table 8. Here lift does not confirm the non-interesting rating given by cosine. As before, *!DP* means that the resource *DP* has not been viewed.

Table 8. Association rules for the other resources

rule	sup.	conf.	cos.	lift
!DP, !AtLeast1Ex → !Book	0.32	0.92	0.62	1.29
Book, DP, AtLeast1Ex → Jflap	0.12	1	0.51	2.08

Keeping in mind the meaning of measures can help deciding what to do with an association.

Let us consider the last rule of Table 6. Cosine indicates that, among the students who have not done *Ex5* nor *Ex6*, over 60% had done Ex1 (consult the typical figures in Table 1), while lift indicates that the proportion of students who have not done *Ex5* and *Ex6* is not larger in students who did *Ex1* (which represented 43% of the students, according to table 2) than in all students. However, from a pedagogical point of view, the case of students who did not attempt *Ex1* is not relevant for this analysis. Therefore the teacher would probably find it useful to keep this rule, hence following cosine, though lift gives here an interesting complementary information.

Let us now consider the second rule of Table 8. Cosine gives us the following information: among the students who consulted the *Book* web site, the extra material on Design Patterns (*DP*) and done at least one exercise, less than 40% used *Jflap* (refer to the typical values in Table 1). The lift gives us the following information: the proportion of students who looked at *Jflap* is higher among the students who looked at the *Book* web site, the *DP* material and done at least one exercise than in the whole student population. Given the very small number of students (as we can see from the support, there are 10 students who satisfied these three criteria on the left hand side of the rule), it is prudent to follow cosine and reject the rule. It is interesting to note though that if the cosine and lift had given similar values, but with a higher number of students satisfying the three criteria in the left side of the rule, it would then have been advisable to follow the lift and retain the rule.

Conclusion

Association rules are useful in Educational Data Mining for analysing learning data. This technique requires not only that adequate thresholds be chosen for the two standard parameters of support and confidence, but also that appropriate measures of interestingness be considered to retain meaningful rules and filter uninteresting ones out. In this paper we revisited and gave an interpretation for two interestingness measures: cosine and added value (which we saw is closely related to the lift). We presented typical values for these measures. An association rule is rated uninteresting by cosine if its value is around or smaller than 0.65, whereas it is rated uninteresting by the lift if its value is around or under 1. We came to a similar conclusion as in [3]: the interestingness of a rule should be first measured by the cosine, then with lift if cosine rated it as uninteresting. In case of conflict between the two measures, the user needs to take into account the intuitive information provided by each measure and decide upon it. The case study

presented in the paper depicts a standard situation: a LMS provides additional resources for students in complement to the face-to-face teaching context. Teachers want to figure out whether students use these resources and whether these have any (positive) impact on marks. Few association rules (without being strong symmetric ones) came out with a contradictory result for cosine and lift. Keeping in mind the intuition behind cosine and lift helped to decide whether to discard these rules.

Another conclusion of this work is that common LMS are far from being data mining friendly. Log data concerning access to resources and test data are not stored the same way for example. Complex data manipulation is needed to get all data consolidated in a useful form. LMS present statistics, however these are very limited. LMS should be enhanced with a special module with good facilities for exploring data. Data mining tools for LMS should have an association rules module with good facilities to choose the attributes to derive association rules for and with the two interestingness measures cosine and lift.

References

[1] Beck J. Difficulties in inferring student knowledge from observations (and why you should care). *Educational Data Mining workshop, in conjunction with 13th International Conference of Artificial Intelligence in Education,* Marina del Rey, CA. USA. July 2007, pp.21-30 .

[2] Garcia J., Romero C., Ventura S., Calders T. Drawbacks and solutions of applying association rules mining in learning management systems. *Proceedings of the International Workshop on Applying Data Mining in e-learning (ADML'07),* Crete, Greece 2007, p. 13-22.

[3] Han J., Kamber M.,: Concepts and Techniques, Morgan Kaufmann Publishers, 2001.

[4] Introduction to Automata Theory, Languages, and Computation, http: //infolab.stanford.edu/~ullman/ialc.html, last consulted March 01 2008.

[5] Jflap http://www.jflap.org/ last consulted March 01 2008.

[6] Merceron, A. Unterstützung des Erlernens von endlichen Automaten mit Hilfe von Mustern. *Proceedings Wokshop für Modellierung in Lehre und Weiterbildung,* Desel, J. & Gliz, M. (Eds.), Modellierung 2008, Berlin, Germany.

[7] Merceron, A., Yacef, K. Revisiting interestingness of strong symmetric association rules in educational data. *Proceedings of the International Workshop on Applying Data Mining in e-learning (ADML'07),* Crete, Greece 2007, p. 3-12.

[8] Merceron, A., Yacef , K. Educational Data Mining: a Case Study. *Proceedings of Artificial Intelligence in Education (AIED2005),* Amsterdam, The Netherlands, IOS Press, 2005.

[9] Minaei-Bidgoli, B., Kashy, D.A., Kortemeyer, G., Punch, W.F. Predicting student performance: an application of data mining methods with the educational web-based system LON-CAPA. *ASEE/IEEE Frontiers in Education Conference*. 2003. Boulder, CO: IEEE,

[10] Moodle http://moodle.org/ last consulted March. 01 2008.

[11] Tan P.N., Kumar V., Srivastava J. Selecting the Right Interestingness Measure for Association Patterns. *Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* San Francisco, USA, p. 67-76, 2001.

[12] Wang, F. On using Data Mining for browsing log analysis in learning environments. *Data Mining in E-Learning. Series: Advances in Management Information*,, Romero, C., Ventura, S., Editors, WIT press. p. 57-75, 2006.

Improving Contextual Models of Guessing and Slipping with a Truncated Training Set

Ryan S.J.d. Baker, Albert T. Corbett, Vincent Aleven
{rsbaker, corbett, aleven}@cmu.edu
Human Computer Interaction Institute, Carnegie Mellon University

Abstract. A recent innovation in student knowledge modeling is the replacement of static estimates of the probability that a student has guessed or slipped with more contextual estimation of these probabilities [2], significantly improving prediction of future performance in one case. We extend this method by adjusting the training set used to develop the contextual models of guessing and slipping, removing training examples where the prior probability that the student knew the skill was very high or very low. We show that this adjustment significantly improves prediction of future performance, relative to previous methods, within data sets from three different Cognitive Tutors.

1 Introduction

Developing accurate models of students' knowledge as they use educational software is valuable for many goals. First, it enables learning systems to respond more accurately to differences in student knowledge, optimizing the amount of practice each student receives on each skill [cf. 9]. Second, estimates of student knowledge are often a useful component in the development of models of more complex behavioral constructs, such as gaming the system [3], which are in turn increasingly used in analyses of learning and motivation [cf. 4,10]. As such, assessments of student knowledge are one of the key building-blocks of educational data mining.

One popular and validated method for modeling students' knowledge is Corbett and Anderson's [9] Bayesian Knowledge Tracing model, which has been used within Cognitive Tutors [cf. 1] for mathematics [1], computer programming [9], and reading skill [7]. This model is statistically equivalent to the two-node dynamic Bayesian network used in many other learning environments [14].

Bayesian Knowledge Tracing keeps a running assessment of the probability that a student currently knows each skill, continually updating that estimate based on student behavior and algorithms derived from Bayes Theorem. In order to do this, Bayesian Knowledge Tracing uses four parameters for each skill, including a probability that the student will "guess" and obtain a correct answer without knowing the skill (**G**) and a probability that the student will "slip" and obtain an incorrect answer even though the student knows the skill (**S**). In Corbett and Anderson [9], these parameters are estimated from data for each skill and are invariant across context (i.e. for a given skill, any student will always have the same probability of slipping, no matter what the situation is).

Recent work has attempted to improve on Corbett and Anderson's original approach. Beck and Chang [6,7] noted that multiple sets of parameters fit performance data equally well within Corbett and Anderson's approach, and introduced a method for selecting a single best set of parameters, using Dirichlet Priors fit across skills. The Dirichlet Priors

method significantly improved fit to data from the Geometry Cognitive Tutor[6], but did not improve fit to data from a Cognitive Tutor for middle school mathematics [2].

In other recent work, Baker, Corbett, and Aleven [2] presented a method for estimating the guess (**G**) and slip (**S**) model parameters contextually. In this approach the model's estimate of the probability that an action is a guess or slip is no longer invariant – instead, it depends on details of the action (such as how much time the action took, and how often the student requested help on the skill in the past). Though the Contextual Guess and Slip approach used about half as many parameters as Corbett and Anderson's original approach and Beck and Chang's Dirichlet Priors approach, it was significantly more successful at predicting student performance within an intelligent tutoring system for middle school mathematics [2] than either of these two earlier approaches.

In this paper, we propose an extension to the Contextual Guess and Slip method. Specifically, we refine the training set used to generate the contextual models of guessing and slipping, in order to make the training set more representative of the situations where these estimates will affect predictions of future student performance. We study this new model both in the same data set as [2] and replicate its effectiveness in two new data sets, from Cognitive Tutors on Geometry and Algebra. This paper's contribution is both in showing that this extension significantly improves the model, and in showing that the Contextual Guess and Slip method improves prediction in multiple intelligent tutors.

2 Bayesian Knowledge-Tracing

As previously mentioned, the Contextual Guess and Slip model of student knowledge is an extension of Corbett and Anderson's [9] Bayesian Knowledge Tracing model. Both models compute the probability that a student knows a given skill at a given time, interpreting data on student performance with a four-parameter model.

In the models' canonical form, each problem step in the tutor is associated with a single cognitive skill. The model assumes that at any given opportunity to demonstrate a skill, a student either knows the skill or does not know the skill, and may either give a correct or incorrect response (help requests are treated as incorrect by the model). A student who does not know a skill generally will give an incorrect response, but there is a certain probability (called **G**, the Guess parameter) that the student will give a correct response. Correspondingly, a student who does know a skill generally will give a correct response, but there is a certain probability (called **S**, the Slip parameter) that the student will give an incorrect response. At the beginning of using the tutor, each student has an initial probability ($\mathbf{L_0}$) of knowing each skill, and at each opportunity to practice a skill the student does not know, the student has a certain probability (**T**) of learning the skill, regardless of whether their answer is correct.

However, the parameter values are different between models. In Corbett and Anderson's approach and Dirichlet Priors, each skill has four parameter values, used in all situations. In the Contextual Guess and Slip model, each skill has two parameter values used in all situations ($\mathbf{L_0}$**, T**), and two parameter values which vary according to context (**G, S**).

The system's estimate that a student knows a skill is continually updated, every time the student gives a first response (a correct response, error, or help request) to a problem step. First, the system re-calculates the probability that the student knew the skill before making the attempt, using the evidence from the current step. Then, the system accounts for the possibility that the student learned the skill during the problem step. The equations for these calculations are:

$$P(L_{n-1}|Correct_n) = \frac{P(L_{n-1}) * (1 - P(S))}{P(L_{n-1}) * \big(1 - P(S)\big) + \ (1 - P(L_{n-1})) * (P(G))}$$

$$P(L_{n-1}|Incorrect_n) = \frac{P(L_{n-1}) * P(S)}{P(L_{n-1}) * P(S) + \ (1 - P(L_{n-1})) * (1 - P(G))}$$

$$P(L_n|Action_n) = \ P(L_{n-1}|Action_n) + \big((1 - P(L_{n-1}|Action_n)\big) * P(T))$$

The simplest baseline approach to fitting a Bayesian Knowledge Tracing model is to allow each of the four parameters to take on any value between 0 and 1. Corbett and Anderson [9] instead used a bounded approach, where the guess and slip parameters are not allowed to rise above pre-chosen thresholds. Beck and Chang [7] showed that both of these approaches are prone to the "identifiability problem", where multiple models can fit the data equally well. They proposed that models be chosen using Dirichlet Priors, which chooses a single best model by biasing parameters towards values that fit the whole data set well. Within this paper, we fit parameters for the Dirichlet Priors approach using Bayes Net Toolkit-Student Modeling (BNT-SM) [6].

However, the baseline and Dirichlet Priors approaches may result in parameters which are "theoretically degenerate" [2]. The conceptual idea behind using Bayesian Knowledge Tracing to model student knowledge is that knowing a skill generally leads to correct performance, and that correct performance implies that a student knows the relevant skill. A model deviates from this theoretical conception, and thus is theoretically degenerate, when its guess (**G**) parameter or slip (**S**) parameter is greater than 0.5. A slip parameter over 0.5 signifies that a student who knows a skill is more likely to answer incorrectly than correctly; similarly, a guess parameter over 0.5 signifies that a student who does not know a skill is more likely to answer correctly than incorrectly.

3 The Contextual Guess and Slip Model of Student Knowledge

Baker, Corbett, and Aleven [2] proposed a new way of fitting parameters: estimating whether each individual student response is a guess or a slip based on contextual information (such as prior history and the speed of response), rather than using fixed guess and slip probability estimates across situations. This modeling approach was tested within a data set from an intelligent tutor for middle school mathematics, and significantly reduced the degree of model degeneracy. This approach was significantly better at predicting student performance than models developed using the Dirichlet Priors, bounded, and baseline methods, despite using substantially fewer parameters.

The first step of the Contextual Guess and Slip method is to label a set of existing student actions with the probability that these actions involve guessing or slipping, using the Dirichlet Priors skill estimates. The set of student actions to be labeled is drawn (in this

approach) from the set of first actions on each problem step, on the set of skills for which the Dirichlet Priors model is not theoretically degenerate. This set of skills was used, rather than all skills, in order to avoid training the models to include model degeneracy. Each student action (***N***) is labeled with the probability that it represents a guess or slip, using information about the two actions afterwards (***N+1***, ***N+2***). Using information about future actions gives considerable information about the true probability that a student's action at time ***N*** was due to knowing the skill – if actions ***N, N+1***, and ***N+2*** are all correct, it is (in most cases) unlikely that ***N***'s correctness was due to guessing. The probability that the student guessed or slipped at time ***N*** (i.e., the action at time ***N,*** which we term $\boldsymbol{A_n}$) is directly obtainable from the probability that the student knew the skill at time ***N***, given information about the action's correctness:

$$P(A_n \text{ is guess} \mid A_n \text{ is correct}) = 1 - P(L_n) \qquad P(A_n \text{ is slip} \mid A_n \text{ is incorrect}) = P(L_n)$$

Next, the probability that the student knew the skill at time ***N*** can be calculated, given information about the actions at time ***N+1*** and ***N+2*** (which we term $\boldsymbol{A_{+1+2}}$). This is done by using Bayes' Rule to combine 1) the probability of the actions at time ***N+1*** and ***N+2*** ($\boldsymbol{A_{+1+2}}$), given the probability that the student knew the skill at time ***N*** ($\mathbf{L_n}$); 2) the prior probability that the student knew the skill at time ***N*** ($\mathbf{L_n}$); and 3) the initial probability of the actions at time ***N+1*** and ***N+2*** ($\boldsymbol{A_{+1+2}}$).

In equation form, this gives: $P(L_n \mid A_{+1+2}) = \frac{P(A_{+1+2} \mid L_n) * P(L_n)}{P(A_{+1+2})}$

The probability of the actions at times ***N+1*** and ***N+2*** is computed as

$$P(A_{+1+2}) = P(L_n) * P(A_{+1+2} \mid L_n) + (1 - P(L_n)) * P(A_{+1+2} \mid \sim L_n)$$

The probability of the actions at time ***N+1*** and ***N+2,*** in the case that the student knew the skill at time ***N*** ($\mathbf{L_n}$)***,*** is a function of the probability that the student guessed or slipped at each opportunity to practice the skill. C denotes a correct action; ~C denotes an incorrect action (an error or help request).

$$P(A_{+1+2} = C, C \mid L_n) = P(\sim S)^2 \qquad P(A_{+1+2} = C, \sim C \mid L_n) = P(G)P(\sim S)$$

$$P(A_{+1+2} = \sim C, C \mid L_n) = P(G)P(\sim S) \qquad P(A_{+1+2} = \sim C, \sim C \mid L_n) = P(G)^2$$

The probability of the actions at time ***N+1*** and ***N+2,*** in the case that the student did not know the skill at time ***N*** ($\mathbf{L_n}$)***,*** is a function of the probability that the student learned the skill between actions ***N*** and ***N+1***, the probability that the student learned the skill between actions ***N+1*** and ***N+2***, and the probability of a guess or slip.

$$P(A_{+1+2} = C, C \mid \sim L_n) = P(T)P(\sim S)^2 + P(\sim T)P(T)P(G)P(\sim S) + P(\sim T)^2 P(G)^2$$

$$P(A_{+1+2} = C, \sim C \mid \sim L_n) = P(T)P(\sim S)P(S) + P(\sim T)P(T)P(G)\big(P(S)\big) + P(\sim T)^2 P(G)P(\sim G)$$

$$P(A_{+1+2} = \sim C, C \mid \sim L_n) = P(T)P(S)P(\sim S) + P(\sim T)P(T)P(\sim G)P(\sim S) + P(\sim T)^2 P(\sim G)P(G))$$

$$P(A_{+1+2} = \sim C, \sim C \mid \sim L_n) = P(T)P(S)^2 + P(\sim T)P(T)P(\sim G)P(S) + P(\sim T)^2 P(\sim G)^2$$

Once the set of actions is labeled with estimates of whether each action was a guess or slip, the labels are used to train models that can accurately predict at run-time the probability that a given action is a guess or slip. The original labels were developed using future knowledge, but the machine-learned models predict guessing and slipping using only data about the action itself and events before the action (i.e. no future data is used).

For each action, a set of 23 features are distilled to describe that action, including information on the action itself (time taken, type of interface widget) and the action's historical context (for instance, how many errors the student had made on the same skill in past problems). Linear Regression is then used, within Weka [16], to create 2 models predicting the probability of guessing (model 1) and slipping (model 2).

Finally, these 2 models are used within Bayesian Knowledge Tracing to dynamically estimate the probability that each response is a guess or a slip. The first action of each opportunity to use a skill is labeled (using the machine-learned models) with predictions as to how likely it is to be a guess or slip, and parameter values are fit for P(**T**) and P($\mathbf{L_0}$), for each skill. At this point, this model – like the earlier work – can make a prediction about student knowledge each time a student attempts to use a skill for the first time on a given problem step. It is worth noting that this model involves considerably fewer parameters than previous models – whereas the Dirichlet Priors and baseline models had exactly 4 parameters per skill, this model fits just over 2 parameters per skill (parameters for **T** and $\mathbf{L_0}$ for each skill, with parameters for **G** and **S** amortizes across all skills).

4 Choice of Data Set Used to Train Contextual Models

In the version of the Contextual Guess and Slip method published in [2], the data set used to train a knowledge model is the set of first actions on each problem step, on the set of skills for which the Dirichlet Priors model is not theoretically degenerate. However, there are potential drawbacks to using this data set. Specifically, if the data set involves significant amounts of over-practice, there may be a large number of actions for which a student has a probability close to 1 of knowing the relevant skill. On these actions, the estimated probability that any incorrect response is due to a slip may be very close to 1, and the probability that any correct response is due to a guess may be very close to 0.

To give an example: Let us consider a skill which has Dirichlet Prior values of P(**G**) = 0.3 P(**S**) = 0.2, P(**T**) = 0.1, and at the current opportunity to practice the skill P($\mathbf{L_{n-1}}$) = 0.99. If the current action is incorrect (**~C**), and the following two actions are not correct (**~C,~C**), it is reasonable to assume that the current incorrect action is due to not knowing the skill, rather than a slip. However, the probability that the current action was a slip will be very high, 97.6%, according to the equations above, because of the very high value of P($\mathbf{L_{n-1}}$). This may be the correct prediction in this context; but if the model trains on this prediction and then uses it in different contexts when P($\mathbf{L_{n-1}}$) is further from 1, the probability that those actions are slips may be overestimated. (One explanation for why three errors in a row could occur on a skill with very high P($\mathbf{L_{n-1}}$) is that the mapping between actions and skills may have errors [cf. 8,9]; fixing such errors is a research topic in its own right [cf. 5,8]).

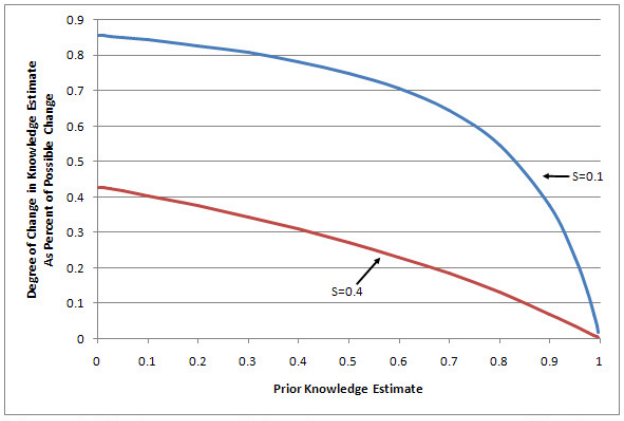


Figure 1. The degree to which an incorrect response can affect the knowledge estimate, for different levels of student prior knowledge and slip (S) parameters. G and T are held constant. The graph of how correct responses can affect the knowledge estimate is similar but reversed horizontally.

Pragmatically, it is more important for these estimations to be accurate when ($\mathbf{L_n}$) is distant from 0 and 1. As P($\mathbf{L_{n-1}}$) approaches 1, P(**S**) has less and less impact on P($\mathbf{L_n}$) – the base probability is too extreme. This can be seen in Figure 1. Similarly, as P($\mathbf{L_{n-1}}$) approaches 0, P(**G**) has less and less impact on P($\mathbf{L_n}$). Hence, it is more important for the model to be highly accurate in cases where P($\mathbf{L_n}$) is not very close to 0 or 1.

One way to accomplish this is to truncate the training set, so that actions where P($\mathbf{L_{n-1}}$) is too close to 0 or 1 are omitted. We choose the cutoffs 0.1 and 0.9, to err on the side of truncating too much rather than truncating too little. Hence, only cases where $0.1 < P(\mathbf{L_{n-1}}) < 0.9$ are included in the training set for the models of guessing and slipping. We can then follow the procedure given in the previous section to create the machine learned models of guessing and slipping, and then use these models in the model of student knowledge.

We call the resultant knowledge model Truncated Training Set Contextual Guess and Slip, or Contextual-Trunc for short. In the following sections, we will compare this model to a version of the Contextual model without any truncation of the training set, and to the Dirichlet Priors model. To avoid bias, all models are evaluated on non-truncated data.

5 Data

We evaluate the models of knowledge tracing discussed here within data sets drawn from three Cognitive Tutors, on Algebra, Geometry, and Middle School mathematics. 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 chosen based on the student knowledge model [1], on-demand help, and instant feedback. Cognitive Tutors have been shown to significantly improve student performance on standardized exams and tests of problem-solving skill [13].

The Algebra and Geometry data sets were obtained from the Pittsburgh Science of Learning Center DataShop (https://learnlab.web.cmu.edu/datashop/). The DataShop is a public resource for the learning science community, giving free access to anonymized

Table 1. The size of each data set (after exclusion of actions not labeled with skills)

	Actions	Problem Steps	Skills	Students
Middle School	581,785	171,987	253	232
Algebra	436,816	136,408	88	59
Geometry	244,398	32,997	144	88

data sets of student use of learning software. The Middle School data set was previously collected by the authors [cf. 3]. Each data set consisted of an entire year's use of an intelligent tutor in schools in the suburbs of a city in the Northeastern USA; we are not aware of any overlap in the student population between data sets. Within each data set, actions which were not labeled with skills (information needed to apply Bayesian Knowledge Tracing) were excluded. However, all other actions on all other skills (including actions eliminated from the Contextual and Contextual-Trunc training sets) are included. The magnitude of the data sets is shown in Table 1.

6 Results

Bayesian Knowledge-Tracing models make predictions about student knowledge (i.e. the probability a student knows a skill at a given time). These predictions can be validated by comparing them to future performance in two ways. The first is to compare actions at time ***N*** to the models' predictions of the probability that actions at time ***N*** will be correct – $P(\mathbf{L_n})*P(\mathbf{\sim S})+ P(\mathbf{\sim L_n})*P(\mathbf{G})$. This method accurately represents exactly what each model predicts; however, this method biases in favor of the Contextual Guess and Slip models, since those models use information associated with the answer being predicted to estimate the probability of guessing and slipping. Therefore, we instead compare actions at time ***N*** to the models' predictions of the probability that the student knew the skill at time ***N***, before the student answered. This method under-estimates goodness of fit for all models (since it does not include the probability of guessing and slipping when answering), but is preferable because it does not favor any model.

We use A' (the probability that the model can distinguish a correct response from an incorrect response) as the measure of goodness-of-fit. A model with an A' of 0.5 performs at chance, and a model with an A' of 1.0 performs perfectly. To assess the statistical significance of the differences between models, we compute A' for each student in each model, compute the standard error of the A' estimates [12], use a Z test to find the difference between models within each student [11], use Stouffer's Z [15] to aggregate across students, and finally compute the (two-tailed) statistical significance of the Z score obtained. This method does not collapse across any data (i.e. it is not overly conservative) but accounts for the non-independence of actions within a single student.

Within the Middle School data set, the Dirichlet Priors approach achieves an average A', across students, of 0.641. The Contextual approach achieves an average A' of 0.749. The Contextual-Trunc approach achieves an average A' of 0.758. The Dirichlet Priors approach is statistically significantly poorer than the other two approaches, Z=59.56,

Table 2. The A' of each model within each tutor, across students. The Contextual-Trunc model is in boldface where it is statistically significantly better than the Dirichlet Priors model, and in italics where it is statistically significantly better than the Contextual model.

	Dirichlet Priors	Contextual	Contextual-Trunc
Middle School	0.641	0.749	***0.758***
Algebra	0.694	0.632	***0.707***
Geometry	0.638	0.666	**0.669**

p<0.0001, Z=64.17, p<0.0001. The Contextual-Trunc approach is statistically significantly better than the Contextual approach, Z=4.59, p<0.0001.

Within the Algebra data set, the Dirichlet Priors approach achieves an average A' of 0.694. The Contextual approach achieves an average A' of 0.632. The Contextual-Trunc approach achieves an average A' of 0.707. The Contextual-Trunc approach is statistically significantly better than the Dirichlet Priors approach, Z=2.89, p<0.01. However, the Contextual approach is statistically significantly worse than the Dirichlet Priors approach, Z= -27.76, p<0.0001. The Contextual-Trunc approach is statistically significantly better than the Contextual Approach, Z= 30.65, p<0.0001.

Within the Geometry data set, the Dirichlet Priors approach achieves an average A' of 0.638. The Contextual approach achieves an average A' of 0.666. The Contextual-Trunc approach achieves an average A' of 0.669. The Contextual-Trunc approach is statistically significantly better than the Dirichlet Priors approach, Z=2.52, p=0.01; the difference between the Dirichlet Priors approach and the Contextual approach is (at best) marginally significant, Z=1.60, p=0.11. The difference between the Contextual and Contextual-Trunc approaches is not significant, Z=0.92, p=0.35.

The full pattern of results is shown in Table 2. As can be seen, the Contextual-Trunc model consistently performed better than the Dirichlet Priors model. The Contextual model, by contrast, performed almost as well as the Contextual-Trunc model in two cases, but was far worse than the other models in the Algebra data set. The primary difference appears to have been that the Algebra Contextual model predicted massively more slips than the other two models did. Whereas the average value of P(**S**) (across skills) in the Algebra Dirichlet Priors model was 0.19, and the average value of P(**S**) (across actions) in the Algebra Contextual-Trunc model was 0.38, the average value of P(**S**) (across actions) in the Algebra Contextual model was 0.67. Values of the slip parameter above 0.5 are degenerate, as discussed earlier; these values cause the model to very quickly infer that a student has mastered a skill, even when the student displays poor performance. By truncating the data set used to train the contextual model of slipping, the Contextual-Trunc model avoids this degenerate performance and is significantly more successful at predicting student performance.

7 Conclusions

In this paper, we have presented an improvement to the Contextual Guess and Slip model proposed in [2]. Earlier models of student knowledge [cf. 7,9] estimated a single probability of guessing and slipping for each skill, and used that estimate for all actions. By contrast, the model presented here (and the model in [2]) contextually estimate the probability that a student obtained a correct answer by guessing, or an incorrect answer by slipping. The Contextual models also use fewer parameters to estimate student knowledge than previous models.

In earlier work [2], contextual models of guess and slip were trained using every action involving non-degenerate skills. In this paper, we adjusted the training set, removing

actions where the probability that the student already knows the skill is below 0.1 or above 0.9. Truncating the training set in this fashion avoids training on cases where probabilities of guess or slip are close to 0 or 1 due to prior probabilities rather than the information contained in successive actions.

We show that using a truncated training set leads to models which are statistically significantly better at predicting future student performance than the Dirichlet Priors approach to parameter selection. A non-truncated training set is also better than Dirichlet Priors in two cases, but in a third case (the Algebra data set) performs significantly worse, due to assigning degenerate values for the slip parameter. This shows that it is valuable to test new student modeling methods on data sets from different learning software (increasingly available in publicly accessible databases such as the PSLC DataShop), since the non-truncated data set would have been perfectly adequate in the Geometry and Middle School data sets.

Further investigation of how to optimally truncate training sets is probably warranted. The choice of 0.1 and 0.9 as cut-offs in this data set is based on data but ultimately arbitrary, and while the solution is effective, a more principled method for selecting cut-offs may lead to better performance. Studying whether truncation of training sets is useful to other classification problems in educational data is another area for future work; input probabilities very close to 0 or 1 are likely to bias the output of any Bayesian method.

At this point, contextual estimation of guess and slip has proven to be better at predicting future performance than earlier methods for student knowledge modeling, for three different learning systems. In the long term, more sensitive and accurate estimation of student knowledge has the potential to improve the effectiveness of learning software. Additionally, as accurate knowledge modeling is a key component of models of complex student behavior used in data mining analyses [cf. 4, 10], better knowledge modeling is likely to be useful to the broader advancement of the field of educational data mining.

8 Acknowledgements

We would like to thank Project LISTEN and Joseph Beck for offering the BNT-SM toolkit used within our model creation process. This work was funded by NSF grant REC-043779 to "IERI: Learning-Oriented Dialogs in Cognitive Tutors: Toward a Scalable Solution to Performance Orientation", and by the Pittsburgh Science of Learning Center, National Science Foundation award SBE-0354420.

9 References

[1] Anderson, J.R., Corbett, A.T., Koedinger, K.R., and Pelletier, R. Cognitive Tutors: Lessons Learned. *Journal of the Learning Sciences*, 1995, 4 (2), 167-207.

[2] Baker, R.S.J.d., Corbett, A.T., Aleven, V. (to appear) More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. To appear in *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*. Online at http://www.cs.cmu.edu/~rsbaker/BCA2008V.pdf

[3] Baker, R.S.J.d., Corbett, A.T., Roll, I., Koedinger, K.R. (to appear) Developing a Generalizable Detector of When Students Game the System. To appear in *User Modeling and User-Adapted Interaction*. Online at http://www.cs.cmu.edu/~rsbaker/USER475.pdf

[4] Baker, R., Walonoski, J., Heffernan, N., Roll, I., Corbett, A., Koedinger, K. Why Students Engage in "Gaming the System" Behavior in Interactive Learning Environments. *Journal of Interactive Learning Research*, 2008, 19 (2), 185-224.

[5] Barnes, T. The Q-matrix Method: Mining Student Response Data For Knowledge. *Proceedings of the AAAI 2005 Educational Data Mining Workshop.*

[6] Beck, J. Difficulties in inferring student knowledge from observations (and why you should care). Educational Data Mining: Supplementary Proceedings of the 13th International Conference of Artificial Intelligence in Education, 2007, 21-30.

[7] Beck, J.E., Chang, K.-m. Identifiability: A Fundamental Problem of Student Modeling. *Proceedings of the 11th International Conference on User Modeling,* 2007.

[8] Cen, H., Koedinger, K.R., Junker, B. Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* 2006, 164-175.

[9] Corbett, A.T., Anderson, J.R. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, 1995, 4, 253-278.

[10] Feng, M., Heffernan, N.T., Koedinger, K.R. Looking for Sources of Error in Predicting Student's Knowledge. *Educational Data Mining: Papers from the 2005 AAAI Workshop,* 54-61.

[11] Fogarty, J., Baker, R., Hudson, S. Case Studies in the use of ROC Curve Analysis for Sensor-Based Estimates in Human Computer Interaction. *Proceedings of Graphics Interface (GI 2005)*, 129-136.

[12] Hanley, J.A., McNeil, B.J. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology,* 1982, 143, 29-36.

[13] Koedinger, K. R., Corbett, A. T. Cognitive tutors: Technology bringing learning sciences to the classroom. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences*, 2006, pp. 61-77. New York, NY: Cambridge University Press.

[14] Reye, J. Student Modeling based on Belief Networks. *International Journal of Artificial Intelligence in Education*, 2004, 14, 1-33.

[15] Rosenthal, R., Rosnow, R.L. *Essentials of Behavioral Research: Methods and Data Analysis*, 1991. Boston: McGraw-Hill.

[16] Witten, I.H., Frank, E. *Data Mining: Practical machine learning tools and techniques,* 2005. San Francisco: Morgan Kaufmann.

Using Item-type Performance Covariance to Improve the Skill Model of an Existing Tutor

Philip I. Pavlik Jr.[1], Hao Cen[2], Lili Wu[1], and Kenneth R. Koedinger[1]
{ppavlik, hcen}@andrew.cmu.edu, lili@cs.cmu.edu, and koediger@cmu.edu
[1]Human Computer Interaction Institute, Carnegie Mellon University
[2]Machine Learning Department, Carnegie Mellon University

Abstract. Using data from an existing pre-algebra computer-based tutor, we analyzed the covariance of item-types with the goal of describing a more effective way to assign skill labels to item-types. Analyzing covariance is important because it allows us to place the skills in a related network in which we can identify the role each skill plays in learning the overall domain. This placement allows more effective and automatic assignment of skills to item-types. To analyze covariance we used POKS (partial order knowledge structures) to analyze item-type outcome relationships and Pearson correlation to capture item-type duration relationships. Hierarchical agglomerative clustering of these item-types was also performed using both outcome and duration covariance patterns. These analyses allowed us to propose improved skill labeling that removes irrelevant item-types, clusters related types, and clarifies the optimal temporal ordering of these clusters during practice.

1 Carnegie Learning's Bridge to Algebra Cognitive Tutor

Our goal was to examine a large dataset (>9 million problems step performances) to determine a skill model that we could subsequently use to produce improvements to a Cognitive Tutor [1]. While the tutor had a skill model coded by human domain experts, we felt that it would be useful to develop alternative methods of coding skills that might be less vulnerable to the possibility of human error and bias. The dataset was provided by Carnegie Learning Inc. from the Bridge to Algebra Cognitive Tutor for the 2006-2007 school year [2]. This tutor works by providing a systematic coverage with 44 units of pre-algebra content each of which has several sections. These sections consist of a problem type, which is composed of several steps or "item-types" which repeat over a sequence of similar problems.

Data from this tutor included times of problem step actions and outcomes of step actions. Because, the human skill model was coded at this step level, we also choose to examine student performances at the step level (henceforth these individual steps in problems in sections in units will be called ***item-types***). By using this level of analysis we will be able to qualitatively compare the skill model that the tutor uses with the skill model suggested in our analyses.

2 Areas of improvement

Human coding and selection of skills in a tutor may introduce the following three possible problems which our datamining algorithm addresses. *As we discuss skills, it should be assumed that we are considering skills more generally as latent variables that*

may represent knowledge components important to a concept, rule application, or other learnable proficiency necessary for responding to an item-type.

2.1 Problem of irrelevant skills

This issue refers to labeling an item-type as needing a skill despite it having a weak relation with other proficiencies that the tutor is focused upon. These weak relationships may be due to the fact that the action is either too simple (probability of success near to 1), too hard (probability of success near 0) or that the task is irrelevant to (independent of) the more complex related target proficiencies. In the case of simple tasks, the skill should not be labeled because the item-type should not be included in the tutor since performance is already at/near the desired level and time is wasted in further repetition. In the case of the difficult task we can suppose that some actions should not be included in a tutor because they result in poor learning and frustrate students. In both of these cases, our method will ignore these skills since a skill lacking variance cannot covary. In the case of an independent task it clearly makes little sense to include the item-type in the tutor unless the data suggests it is associated with a target skill of the domain.

2.2 Problem of skill redundancy

This is the issue of labeling two different item-types with two independent skill labels despite strong similarities in the skill involved. When a human decides to label a skill there is always the question whether the action is statistically equivalent with another action in the tutoring system being analyzed. Of course, this is a very difficult problem to judge since it depends on whether variation in the actions is different enough to produce difference in performances that are of practical significance. Because if this human experts may tend to label skills as different despite strong similarities. This leads to a model that tends to have poor intuitions about the how much practice to give a particular skill because when similar item-types are treated as independent the model can make no conclusions about how much practice to give the second one depending on performance with the first one. Thus, a tutor with redundant skills will be forced to give too little or too much practice because it is ignorant of the underlying skill overlap.

2.3 Problem of skill ordering

It is generally thought that a curriculum has a fixed order in which earlier skills form part of later whole skills or target performances, but this order may be difficult to identify. The importance of curriculum order may be due to a general benefit for training part skills before introducing whole skills. Wightman and Lintern [3] have described two ways of splitting a whole task into components tasks: segmentation and fractionation. Segmentation splits the whole task into sequential steps, and fractionation splits the whole task into time-shared parts. Optimal curriculum ordering may require some part-training in either case. With segmented skills, part-training may allow more efficient targeted training that can avoid already learned parts of the whole task. In the case of fractionated skills, part-training may provide the initial skill that enables the learner to successfully practice the whole skill. Because our method can identify prerequisites, we

can hope that it will help us answer questions about the ordering of skills in a principled way.

3 Analysis methods

The analysis began with the creation of response correctness contingency tables (see Section 3.1) and response duration correlations (see Section 3.2) for all pairwise item-type combinations within each subject's data. Following the creation of these relationship matrices, simple agglomerative clustering iterated until no item-type (clusters) had a relationship correlation product (see Sections 3.2 and 3.3) greater than the clustering coefficient. At this point, the POKS tests were computed on the item-type clusters and the prerequisite order graph was created (one link was also removed by the requirement for $r > 0$ duration correlation between POKS linked item-type clusters).

The parameters were as follows: a clustering coefficient of 0.55, a p_c value of 0.35 (this value needed to be so high because of the strict correction of degrees of freedom described in Section 3) and a_c values of 0.15. These parameters were selected to limit the number of links and restrict the clustering to obtain a graph that would provide us examples we could analyze as discussed by Desmarais [4]. Additionally, before analysis, we removed students with less than 250 transactions, and we removed item-types with less than 2000 transactions. These filters reduced our dataset to 1073 students with a total of 1,099,642 outcomes (the quantity of response durations was slightly lower since we only accepted durations for trials with a successful outcome and discarded durations greater than 60s). The analysis was restricted to this subset, which included all transactions from 3 contiguous units of the 44 in the tutor, to make it manageable to show a graphic of the results. Larger numbers of units/item-types could not be made to fit on a single page.

3.1 Partial order knowledge structures (POKS)

To construct the graph describing the prerequisite relationships in the data, we used the POKS method, which is one way to characterize the theory of knowledge spaces. Knowledge spaces describe how the learn units (skills or item-types in this paper) in a domain are learned in a constrained order [5]. Knowledge spaces have been investigated by many researchers using different methods [6, 7].

To explain the POKS method, first, we introduces the notations following [4]:

$A, B, \ldots$ the upper case Roman letters denote two item-types in test
$A \Rightarrow B$, knowing how to solve item-type A correctly leads to solving item-type B correctly (we have reversed this arrow notation in Figure 1.)
$P(B \mid A)$, the probability of getting item-type B right, given A was right
$P(\neg A \mid \neg B)$, the probability of getting item-type A wrong, given B was wrong
p_c, the minimum probability that $P(B \mid A)$ and $P(\neg A \mid \neg B)$ need to hold
α_c, the error of the POKS tests, which may be set differently for different tests
$N_{A \wedge B}$, the number of times that students get A right and B right*

$N_{A\wedge\neg B}$, the number of times that students get A right and B wrong*

$N_{\neg A\wedge B}$, the number of times that students get A wrong and B right*

$N_{\neg A\wedge\neg B}$, the number of times that students get A wrong and B wrong*

* Because the independence of observations assumption of the statistical tests was strained when considering repetitions of the same item-type for the same student, these values were normalized by dividing the each by the total so that they summed to 1. The statistical tests then assumed a number of degrees of freedom equal to the number of subjects in each pairwise comparison. This correction is overly conservative, but provides an unbiased correction for the sometimes great between-subjects variability in the *N* of repetitions.

$CDFBinomial(x, n, p)$, the cumulative density function of a binomial distribution of n trials and p success probability

The idea of POKS is that if $A \Rightarrow B$ perfectly, we would expect $P(B \mid A) = 1$, $P(\neg A \mid \neg B) = 1$ and that the contingency table shows a lack of independence . In reality, due to noise and imperfect $A \Rightarrow B$, we would expect the above two equalities not to hold exactly. Thus we can setup tests such that if $P(B \mid A)$ and $P(\neg A \mid \neg B)$ are above some threshold p_c, we can have some confidence of $A \Rightarrow B$. Therefore, for there to exist a relationship between A and B three tests must succeed. The first two tests check that $P(B \mid A)$ and $P(\neg A \mid \neg B)$ are above some threshold p_c, given the allowed test error α_c. The third test verifies whether the conditional probability $P(B \mid A)$ and $P(\neg A \mid \neg B)$ are different from $P(B)$ and $P(\neg A)$.

Test 1 returns true if $CDFBinomial(N_{A\wedge\neg B}, N_{A\wedge B} + N_{A\wedge\neg B}, 1 - p_c) < \alpha_c$.

Test 2 returns true if $CDFBinomial(N_{A\wedge\neg B}, N_{\neg A\wedge\neg B} + N_{A\wedge\neg B}, 1 - p_c) < \alpha_c$.

Test 3 returns true if the 2*2 contingency table of $N_{A\wedge B}$, $N_{A\wedge\neg B}$ $N_{\neg A\wedge B}$ and $N_{\neg A\wedge\neg B}$ passes a χ^2 test with error rate α_c.

3.2 Clustering based on conditional log odds

As we can see by examining the tests, they rely on the contingency table that is tabulated for each pair-wise item-type comparison. These contingency tables create a covariance structure that "places" each item-type in the POKS graph relative to the other item-types. By reflecting on this we can see that if we want to cluster the items based on the similarity of the required proficiencies, which would imply they require the same skills, we need a distance metric for item-types that a) captures that two items co-vary and b) can cope with the fact that two items may not be equally difficult despite having the very similar covariance structures. Requirement *a* means we need a distance metric that captures the structure of the contingency tables for item-type X_1 as compared to the contingency tables for item-type X_2. Requirement *b* means that this metric probably should not capture the structure of the tables relative to the outcome of performance X_1 or X_2. Rather, we should describe a distance metric that is computed conditionally for those

cases where X_1 or X_2 is a success or failure. Requirement b is important for the purpose here because the tutor introduces item-types in a fixed order. This fixed order means differences in average performance between item-types may be caused by learning. However, this difference in performance between item-types that represent the same skill should not greatly alter the contingencies given the response is a success or failure.

To do this comparison of the covariance structure it helps to consider the data for two item-types (X_1 and X_2) as being organized into two vectors of contingency tables describing these item-types relationship with all other possible item-types (Y_n). If we consider that each contingency table is organized with X frequency results for A item-type and Y frequency results for B item-type, then for each X_n by Y_n contingency table we individually computed 2 values: one for when X_n is a success and one for when X_n is a failure. In each of these 2 cases, the log odds of B vs. ~B frequencies is used to capture the strength of the odds B:~B on a continuous scale. Because these log odds do not capture the effect of frequency of A or ~A and only capture the relative frequency of B vs. ~B they are not reactive to learning of A that does not affect the patterns of B vs. ~B, nor are they reactive to difference in the *n* of observations of the B:~B results. Using this procedure we computed these 2 log odds (one for A and one for ~A) for each contingency table for each vector of contingency tables (X_1 or X_2).

At this point we can describe vectors of log odds values for each column item-type X_1 and X_2 (getting 2 values conditional on A and ~A for each item X_n by Y_n pair) and compute their Pearson correlation to determine the nearness of the two item-types in the knowledge space. To do this clustering we used a simple agglomerative hierarchical clustering to cluster item-types into a new grainsize which implies clustered items share the same performance requirements (skill). This new method shares similarities with correlation clustering methods that have proven useful for graph partitioning [7] and is described further in the next section.

3.3 Integrating duration covariance information

Previous work to understand the knowledge space has focused exclusively on how performance success or failure can be used to determine ordered structures. However, besides possessing success data, we also had data on the duration of each item-type performance. This data allowed us to compute pairwise duration correlations (*r* values) of the item-types that correspond with the POKS tests for each pairwise item-type relationship. While it was perhaps possible to use these correlations in some joint function with the strength of the result of the POKS tests for each pair, at this point we just used these values as an additional filter on which POKS implications we accepted as significant. For this paper we choose to exclude any A→B pairs where $r < 0$.

More importantly the duration correlation vectors created for each item-type were themselves correlated to produce values that represented the degree of similarity in the duration relationships between item-types. This statistic for each pair of item-types was multiplied by the correlation from the outcome based (log odds vectors) correlation above, and the item-type pair with the highest correlation product is clustered in each step

of the simple agglomerative clustering. Clustering continues until the correlation of pairwise correlation vectors is above the clustering coefficient.

4 Results

Figure 1 shows the POKS graph obtained from this analysis and corresponds to the groupings in Table 1 which provides additional statistics to help interpret the results. The ovals in Figure 1 represent collections (or individual) item-types which were a function of the clustering procedure (also grouped in Table1). Item-type Label indicates the following information (probability correct_Unit name_section number_Skill ID number). The table also provides the average duration and total number of database observations (Rps—repetitions in 1000s) for each item-type. Colors indicate the majority unit membership for the grouped item-types, where LCM – least common multiple unit, GCF – Greatest common factor unit, and FracRep involves a visual and written fraction representation unit. Edge labels provide the average p_c value and the duration correlation r.

4.1 Irrelevant knowledge components

The analysis failed to find any covariance relationship for 17 of the item-types in the 3 units of the tutor. These knowledge components provide an example of how this method can be used to suggest proficiencies that do not covary with the other item-types in the tutor. These so-called irrelevant skills tend to be knowledge components with higher probability correct because in cases with higher probability correct there is less chance to get the examples of not A and not B that are needed to pass the 2nd binomial test. This means that these items are found to be irrelevant because they are so easy that it is less likely to detect how they influence other item-types even where such relationships might exist given a similar problem with higher difficulty.

While further analysis would be necessary to determine if these skills were truly irrelevant, the method has provided us with an initial hypothesis about which skills might be removed from the tutor so that time saved to spend on item-types with stronger relationships with the other tutor content.

4.2 Redundant knowledge components

The clustering of item-types indicates that the pattern of success contingency tables and the pattern of duration correlations were similar for these item-types such that if item-types X and Y are in a cluster it indicates they have similar relationships to the other item-types. By extension we can suppose that this similar place in the covariance structure suggests that performance for these item-types is constrained by the same skill. The fact that this clustering occurs suggests that the human coders used statistically irrelevant features to code the item-types. For example consider the green ovals in Figure 1. The right oval includes a variety of item-types that might be described as understanding the denominator, while the left oval includes item-types that deal with the numerator.

Table 1. Key for graph.

Item-type Label	Rps	Duration	Unit	Sect	ID	Skill description
0.97_FracRep_1_43,35	2k	4.8s	FracRep	1	43,35	Identify benchmark fraction, Identify Fraction using shape
0.91_FracRep_1_46	5k	7.3s	FracRep	1	46	Identify fraction associated with each piece of a horizontal bar
0.92_FracRep_1_35,34	22k	6.2s	FracRep	1	35,34	Identify Fraction using shape, Identify non-benchmark fraction
0.87_FracRep_1_40	5k	7.7s	FracRep	1	40	Identify fraction associated with each piece of a circle
0.85_FracRep_1_38	6k	8.1s	FracRep	1	38	Identify fraction associated with each piece of a square
0.91_FracRep_4_55	4k	5.4s	FracRep	4	55	Identify number of desired groups—construct
0.88_FracRep_5_56,34	4k	14.1s	FracRep	5	56,34	Identify fraction using number line, Identify non-benchmark
0.93_FracRep_3_53	7k	6.8s	FracRep	3	53	Identify fraction of desired items
0.96_FracRep_1_47	2k	5.6s	FracRep	1	47	Count number of shaded parts in square (discontiguous)
0.94_FracRep_1_42	2k	5.7s	FracRep	1	42	Count number of shaded parts in circle (contiguous)
0.95_FracRep_1_49	2k	7.0s	FracRep	1	49	Count number of shaded parts in circle (discontiguous)
0.95_FracRep_1_44	2k	5.6s	FracRep	1	44	Count number of shaded parts in horizontal bar (contiguous)
0.87_FracRep_1_37	3k	7.5s	FracRep	1	37	Count number of shaded parts in square (contiguous)
0.9_GCF_2_28	28k	9.5s	GCF	2	28	List factor of large number
0.94_FracRep_2_40	3k	6.7s	FracRep	2	40	Identify fraction associated with each piece of a circle
0.79_FracRep_3_52	4k	8.4s	FracRep	3	52	Identify number of total items
0.76_FracRep_6_61,59	6k	5.8s	FracRep	6	61,59	Represent non-benchmark, Represent fraction using num line
0.89_FracRep_6_58	5k	6.3s	FracRep	6	58	Identify fraction associated with each segment of a number line
0.7_FracRep_1_32	7k	8.8s	FracRep	1	32	Identify fraction associated with each piece of a vertical bar
0.96_FracRep_2_45	3k	24.0s	FracRep	2	45	Identify number of equal divisions (horizontal bar)
0.94_FracRep_2_33	3k	25.8s	FracRep	2	33	Identify number of equal divisions (vertical bar)
0.86_FracRep_2_41	3k	23.3s	FracRep	2	41	Identify number of equal divisions (circle)
0.93_FracRep_2_39	3k	23.6s	FracRep	2	39	Identify number of equal divisions (square)
0.88_FracRep_6_57	5k	18.3s	FracRep	6	57	Identify number of equal divisions (number line)
0.75_FracRep_5_57	7k	22.2s	FracRep	5	57	Identify number of equal divisions (number line)
0.83_FracRep_1_33	6k	23.2s	FracRep	1	33	Identify number of equal divisions (vertical bar)
0.91_FracRep_1_45	6k	23.6s	FracRep	1	45	Identify number of equal divisions (horizontal bar)
0.87_FracRep_1_41	6k	23.0s	FracRep	1	41	Identify number of equal divisions (circle)
0.85_FracRep_4_54	4k	21.0s	FracRep	4	54	Identify number of equal groups from fraction
0.93_FracRep_1_39	6k	22.6s	FracRep	1	39	Identify number of equal divisions (square)
0.93_LCM_1_21	3k	7.3s	LCM	1	21	Identify LCM - is product
0.85_LCM_1_25	5k	9.7s	LCM	1	25	Identify LCM
0.82_LCM_1_26	5k	8.5s	LCM	1	26	Identify LCM - one number multiple of other
0.91_GCF_1_30	8k	6.9s	GCF	1	30	Identify GCF
0.87_GCF_1_29	9k	7.0s	GCF	1	29	Identify GCF - one number multiple of other
0.91_LCM_1_23	145k	5.5s	LCM	1	23	List consecutive multiples of a number
0.77_LCM_2_21	5k	10.6s	LCM	2	21	Identify LCM - is product
0.7_LCM_2_26	7k	9.2s	LCM	2	26	Identify LCM - one number multiple of other
0.68_LCM_2_25	9k	11.4s	LCM	2	25	Identify LCM
0.69_GCF_2_29	10k	15.2s	GCF	2	29	Identify GCF - one number multiple of other
0.66_GCF_2_30	11k	15.2s	GCF	2	30	Identify GCF
0.56_GCF_2_31	48k	10.7s	GCF	2	31	Identify number of items in each group from GCF

Much of this clustering may be necessary because the human coders were instructed to code in as fine a grain as practical. This instruction led to different skills being coded depending on whether the stimulus was a vertical bar, horizontal bar, circle, square, or number line. In contrast, the clustering method lumped these skills together indicating they may be actually the same proficiency. By splitting these groups into separate skills the human coder delinked these proficiencies relative to the tutor's automatic scheduling mechanisms. So, for example, if a student does very well on these clustered item-types as they are introduced, it will not result in less practice for the other items that our analysis suggests are in the cluster. Therefore by proposing these clusters we can address learning of the concept more efficiently because we can model transfer between item-types that are controlled by the same underlying proficiencies. Modeling transfer between item-

types allows us to know when a particular concept, skill or procedure has been mastered despite the fact that we may not have given a student examples of all the item-types in the cluster. (Also note that sometimes the human coders did repeat the same skill IDs for isomorphic item-types in different sections of the same unit. As we can see in Table 1, our clustering method tended to confirm these human skill labels by clustering these item-types. E.g. skill id 33 (and others) appear twice in the same cluster indicating that the model agrees with human coders decision to label these item-types with the same skill despite the fact that they are in different sections of the same unit.)

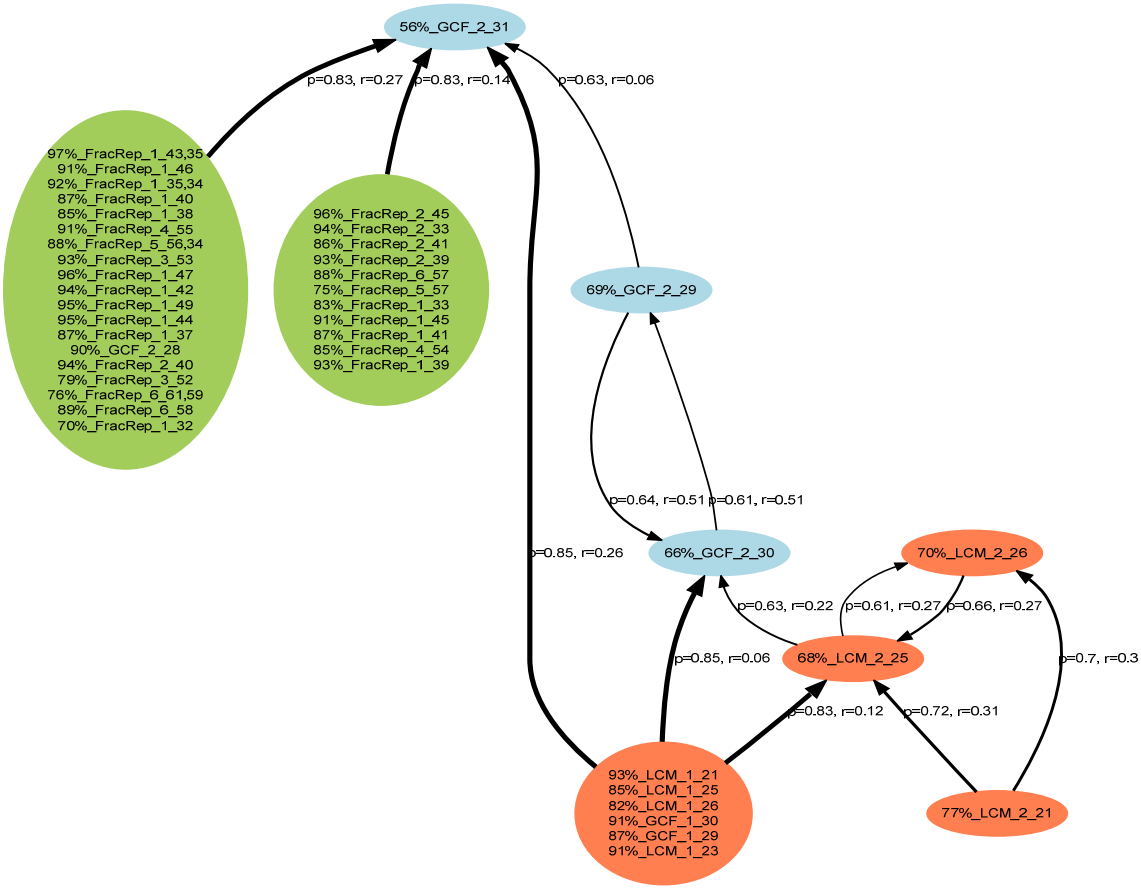


Figure 1. Graph structure described in the results section.

4.3 Ordering of knowledge components

The data comes from a tutor where the units follow a fixed order, and we can use our analysis to question the appropriateness of that order. As discussed in the introduction, we assume that introducing a prerequisite before its post-requisite will result in better learning, because each new idea will have been more adequately prepared by the scaffolding from prerequisite practice. This analysis of the optimal order is more difficult (than analysis of clustering or irrelevant skills) because the tutor repeats item-types, and learning caused by this repetition might explain why a downstream item-type performs better than an earlier item-type. However, while ideally we would include both orders of performance of any pair of item-types in our sample, it still seems safe to infer that very

strong prerequisite relationships are not determined mostly by learning effects. Take for instance the position of the two green clusters (fraction concepts) relative to the GCF (greatest common factor unit)_2_31 item-type. Skill ID 31 involves a word problem in which students must produce the other factor for each of 2 products when the first factor has just been supplied by the student, e.g. “You have groups of 4 apples and 6 pears, what is the greatest number of equal sized groups of fruit you can make? (This is an ID 30 skill.) How many apples in each group? (ID 31) How many pears in each group? – (also ID 31)”. This dependence of skill ID 31 in section 2 of the GCF unit on the fraction concept clusters seems plausible since this contextualized problem involves the denominator concept of understanding that wholes can be divided and also the numerator concept that these portions must be composed of a certain count of parts. While this reasoning might normally seemed strained, the support from the graph implies that the FracRep item-types should be practiced before this contextualized GCF section if we want to respect the recommendations of the theory of part-whole training to address the prerequisite skill first.

5 Conclusions

Future work will focus on integrating this knowledge space analysis with tracking of individual skills such as is currently used in the Bridge to Algebra Tutor. By integrating the knowledge space analysis it appears that we can get a rich perspective on what student actions might deserve to be coded as independent knowledge components. As we discussed in the results, this perspective should improve the performance of the model that tracks repetition of single skills in the tutor because that model can be modified to remove irrelevant skills, made less redundant by clustering skills, and made to better conform to the theory that prerequisites should be trained before later skills.

This integration may proceed as shown in work by Cen on the Learning Factors Analysis (LFA) method, which allows improvement in Cognitive Tutor models by searching a space of hypothetical skills for the combination that best fits previously collected data [8]. LFA starts with an initial cognitive model represented as a binary matrix that maps a collection of skills to each item-type (or item) and uses a set of customized item response models to evaluate the model fit produced by any given mapping of skills to item-types for a particular dataset. These binary matrices are based on the tentative judgments of human experts about the effect of the features of the item-types, and LFA can systematically incorporate those features into existing cognitive models by generating and searching for alternative skill labels as allowed for in the matrix. This method has been used by various researchers to evaluate cognitive models in geometry, physics and reading [9, 10] . However, the method still requires a domain expert to propose alternative labeling of skills along which the algorithm searches. The methods proposed in this paper show promising potential to combine the strengths of POKS, item-type clustering and LFA to answer various EDM research questions by allowing us to use POKS and item-type clustering to generate a starting or alternative binary skill matrix for LFA model search.

Acknowledgements

This research was supported by the U.S. Department of Education (IES-NCSER) #R305B070487 and was also made possible with the assistance and funding of Carnegie Learning Inc., the Pittsburgh Science of Learning Center, DataShop team (NSF-SBE) #0354420 and Ronald Zdrojkowski.

References

[1] Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R. Cognitive tutors: Lessons learned. *Journal of the Learning Sciences,* 1995, 4, p. 167-207.

[2] Carnegie Learning Inc. [Bridge to Algebra Cognitive Tutor data for the 2006-2007 school years].

[3] Wightman, D.C., Lintern, G. Part-task training for tracking and manual control. *Human Factors,* 1985, 27, p. 267-283.

[4] Desmarais, M.C., Maluf, A., Liu, J. User-expertise modeling with empirically derived probabilistic implication networks. *User Modeling and User-Adapted Interaction,* 1996, 5, p. 283-315.

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

[6] Falmagne, J.-C., Doignon, J.-P., Cosyn, E., Thiery, N. The assessment of knowledge in theory and in practice. *Institute for Mathematical Behavioral Sciences,* 2003, #26.

[7] Desmarais, M.C., Gagnon, M.. Bayesian student models based on item to item knowledge structures. *First European Conference on Technology Enhanced Learning,* 2006, Crete, Greece.

[8] Cen, H., Koedinger, K., Junker, B. Learning Factors Analysis - A General Method for Cognitive Model Evaluation and Improvement. *8th International Conference on Intelligent Tutoring Systems,* 2006.

[9] Cen, H., Koedinger, K., Junker, B. Is Over Practice Necessary? – Improving Learning Efficiency with the Cognitive Tutor through Educational Data Mining. *13th International Conference on Artificial Intelligence in Education,* 2007, Los Angeles, CA.

[10] Rafferty, A., Yudelson, M.: Applying Learning Factors Analysis to Build Stereotypic Student Models. *13th International Conference on Artificial Intelligence in Education* (Best Student Paper), 2007, Los Angeles, CA.

Data-driven modelling of students' interactions in an ILE

Manolis Mavrikis
m.mavrikis@lkl.ac.uk
London Knowledge Lab, Institute of Education
23-29 Emerald Street, WC1N 3QS, London, UK.

Abstract. This paper presents the development of two related machine-learned models which predict (a) whether a student can answer correctly questions in an ILE without requesting help and (b) whether a student's interaction is beneficial in terms of learning. After presenting the rationale behind the development of these models, the paper discusses how the data collection was facilitated by the integration of different versions of the ILE in realistic classroom situations. The main focus of the paper is the use of the ICS algorithm of WEKA to derive Bayesian networks which provide satisfactory predictions. The results are compared against decision trees and logistic regression. The application of these models in the ILE and how their potential educational consequences were taken into account are outlined followed by a discussion of future lines of research.

1 Introduction

Towards developing components of an Interactive Learning Environment (ILE) to enable the provision of adaptive feedback that takes into account students' cognitive and affective characteristics, several aspects of students' interactions with the ILE require explicit modelling. Although the aspects that require modelling can be identified through qualitative analysis and traditional knowledge elicitation, the actual development of the models can be more valid by employing statistical, data mining and machine learning techniques. This paper focuses on the development of two machine-learned models related to each other. The first one predicts whether students can answer a particular question correctly without further help, given their interaction so far. The second model, provides a measure of how beneficial, in terms of learning, their interaction with the ILE is.

Before presenting the development of the models in detail, the next two sections, describe briefly the aforementioned ILE, called WaLLiS, the rationale behind the development of these models, how they are related, as well as the datasets that were used for the machine-learning task. The rest of the paper presents the models and Section 4, after briefly discussing the application of the models in the system raises issues for future work.

1.1 Context

WaLLiS has been described in detail in [12] and in relation to other studies in [13]. Briefly, it is a web-based environment that hosts contents which include theory or example pages that present the material, as well as interactive and exploratory activities. The overall environment of WaLLiS follows a design that is similar to systems referred to as advanced

eLearning environments, as they combine features of content-based approaches with adaptive educational strategies from Intelligent Tutoring Systems (ITS). Accordingly, apart from the usual components of the system that deliver the material and the tree-based navigation of the content (typical in many eLearning systems), WaLLiS incorporates a feedback frame at the bottom of the window where adaptive feedback is provided to the students.

Despite the fact that several studies with the system established its effectiveness [12], some of the students are interacting with it in ways which are not necessarily beneficial to their learning. It is often the case that the approach followed in this kind of situations is to redesign the system in ways that will prevent any undesirable behaviour [3]. However, this may introduce new problems. For example, in early versions of WaLLiS hint and solution requests were not permitted without first attempting to answer the question. This however, led students to answer randomly just to 'game the system' [3] into allowing them to request hints or solutions. Similar results from designing preventative approaches are described in [14]. Even though newly introduced behaviours can always be dissuaded, as discussed in [3], this leads to an 'arms race' where students are developing harmful (in terms of their learning) behaviours and designers are trying to stop them. On the other hand, it seems that a measure of 'desirable' interaction could empower the system with an indication of the benefit of the interaction which could be used to guide feedback provision without repeatedly redesigning the system's interaction model.

In addition, related literature [7, 15, 19] and a combination of qualitative research and statistical analysis [11, 16] indicate that part of the evidence that human tutors employ, in order to diagnose student affective characteristics (e.g. confidence, effort), comes from students' help-seeking behaviour and particularly help requests for items on which the tutors, based on the quality of previous interactions with the item under question, estimate that a student's request for help is superfluous. This suggests that a first step in predicting affective characteristics and developing a model of *beneficial interaction* is to be able to predict if a student could answer correctly without any *need for help*.

As already mentioned, rather than employing arbitrary models based on intuition, or even expert elicitation, the development of the models was data-driven. The assumption behind learning models from data is that differences in learning style, in affective characteristics and other preferences are reflected in students' interactions with the system.

1.2 Datasets

The collection of as realistic data as possible was facilitated by the iterative design methodology behind the WaLLiS project and the integration of the ILE in the teaching and learning of various courses of the School of Mathematics of the University of Edinburgh and in particular, Geometry Iteration and Convergence (`GIC`); a second year module undertaken by honour students. With the lecturers' agreement, the course was used as a means of conducting studies. Materials were developed for one of the last concepts taught in GIC; *conic sections*. The main reason for choosing this particular part of the course was that the materials taught were unknown to the students in advance and they constitute a rather individual unit. In addition, it was possible to establish indicators of prerequisite knowledge and opportunities to deliver part of the course solely through WaLLiS. Moreover, one of the activities (converting a conic section into its standard form), which is used for most

of the analysis in relation to help-seeking and students' performance, was presented using a different method from course textbooks. This helped establishing that any performance results are reasonably (if not solely) attributed to the interaction with the system and not other external factors. In particular, with the collaboration of the lecturer, certain questions on the students' final exam were designed to test long-term retention.

Following an iterative design and after several successful pilot tests, three datasets were particularly useful for the machine learning analysis described in this paper. `GIC03` was the first dataset collected from a formal application of WaLLiS in classroom as the sole means of teaching conic sections. The main reason behind this data collection was to perform a qualitative analysis of the way students interact and perceive the system and not to focus on learning outcomes. 126 students interacted with the system. `GIC04` (133 students) and `GIC05` (115 students) aimed particularly at collecting post-test results. Learning gains could be assessed by averaging (a) the students' marks on an assessment they had to complete right after their interaction and (b) their mark on a specially designed final exam.

It is evident that data collection under realistic conditions entails several challenges that result in ignoring some data, and it is worth mentioning them. Due to the way the datasets were collected (i.e. remotely over the net and not during a lab study), data can be quite noisy. The methods used for data-collection [10] are subject to bandwidth availability, appropriate security settings and other client and server-side concerns. In addition, some students did not give their consent for their data to be recorded. Data from students who did not attend the familiarisation session, and from others who have taken the course in the past were also ignored since their behaviour was quite different. After this data cleaning process the `GIC03`, `GIC04`, and `GIC05` contain 106, 126 and 99 students respectively.

2 Predicting the necessity of help-requests

Knowing whether a student needs help or not in a given educational situation is quite complex. In the context of educational technology this information is particularly crucial for Intelligent Tutoring Systems (ITS) [1] and definitely not a unique issue to research here. Because of its complexity, different researchers address it in different ways depending on the special characteristics of the system under consideration and the overall context. For example, in the CMU tutors (e.g., [2]) the problem is approached as an attempt to estimate the probability that a skill has been mastered (*knowledge tracing* [6]). Similarly, [5, 9, 17] describe systems where Bayesian networks are used to predict students' knowledge during their interaction. The approach presented here is different. The model predicts students' necessity to ask for help on an item given their previous interaction and it is learned based on data of all students' interactions with previous applications of the system in classroom.

As discussed in the previous Section the GIC datasets were collected from studies where students have no previous knowledge of the material. Therefore, it does not seem too bold to assume that students who do not ask for help and answer a question correctly with the first attempt have learnt either from carefully reading the material in the system or from the interaction with the related items. In other words, all other characteristics of a student being equal, similar interactions should have enabled students to answer without the need for help. The opposite is not necessarily true as individual differences between students and affective characteristics influence whether a student requests help or not.

Initial investigations with the GIC03 dataset as a learning set and the GIC04 as a testset, supported the claim that a machine learning algorithm (such as bayesian networks) could be used to automatically predict with reasonable accuracy whether a student's help-requests are necessary. It was decided to focus the prediction only on help requests prior to the first attempt to answer a question. Further attempts are quite complex and depend on students' understanding of the feedback, whether they read it or not and several other factors, which add noise to the prediction. Given the above assumptions and in order to learn a more accurate model from the data both the GIC03 and GIC04 datasets were used as a training set. In an attempt to have a simple model and a method that could be generalised to other courses of WaLLiS or other ILEs, only few aspects of the interaction were considered as features for the learning task. These should be available across courses in WaLLiS and are quite common in ILEs. Accordingly, vectors were constructed that contain the following variables: (a) time spent on related pages (*trp*) (b) time spent on attempt (*tsa*) (c) student previous knowledge (*prev*) (d) a rule-based measurement of the degree of 'completeness' of the goals of interactions on related pages[1] (*rel*) (e) difficulty of the item (*diff*) and (f) the type of the answer required (mcq, blank, matrix) (*answertype*).

The boolean class learned represents whether the student seems to be able to answer correctly without any help. Its value therefore, is FALSE when students provided completely wrong answers (not from the usual misconceptions), or answered wrongly very quickly[2] demonstrating, in a sense, that they only answer in order to 'game' the system into providing feedback. The value of the class is TRUE when a student's answer was correct or partially correct. Students who asked for help without an attempt were not included since there are many explanations behind this request. Using these data for the machine learning would not necessarily provide additional instances that demonstrate whether a student really needed help or not. All the above restrictions resulted in a set of 1230 instances (the class of 429 of which was FALSE). The next step was to choose the exact modelling method. Preliminary investigations with cross-fold validation suggested that from all the approaches attempted (decision trees, Bayesian network, classification via regression) the Bayesian network and the decision tree were the most accurate ones with no significant differences. The Bayesian network (BN) approach was preferred mainly because the nature of the prediction is highly probabilistic. In addition, a BN is the perfect candidate for employing its outcomes in a larger evidence-based probabilistic framework.

In WEKA [20], learning a BN is considered as a learning task of finding an appropriate classifier for a given dataset with a class variable and a vector of attributes [4]. The learning is a two stage process of first finding an appropriate network structure and then learning the probability tables. There are several approaches for learning the structure of the network. *Conditional independence test based structure learning* approaches stem from the need to uncover causal structure in the data [4] and consider the task as an attempt to learn a network structure that represents the independencies in the distribution that generated the data. Although directed edges in a network do not necessarily represent causal effects, the

[1] This was based on intuition and expert knowledge elicitation from the course lecturer.

[2] The time to answer was discretized following a technique similar to the one presented in [8]. The number of breakpoints was chosen empirically in an attempt to maintain the proportionality of a normal distribution and the notion of the fuzzy linguistic variables ("very quickly", "quickly" etc.). Accordingly, the breakpoint for "very quickly" was $z <= -1.28$.

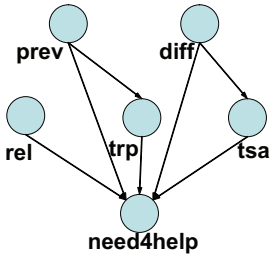


Figure 1. Bayesian network predicting the necessity of help-requests

		BayesNet		J4.8	
		Cross	Test	Cross	Test
	accuracy	67.64	66.52	65.84	64.05
	Kappa	0.32	0.32	0.30	0.23
recall	True	0.74	0.77	0.71	0.72
recall	False	0.61	0.56	0.59	0.50

Figure 2. Accuracy, Kappa statistic and recall values for two different techniques

ICS algorithm [18] as implemented in WEKA [20] starts from a complete undirected graph for each pair of nodes, it considers subsets of nodes that are neighbours to the pair. If an independence is identified, the edge between the pair is removed from the network structure and the arrows are directed accordingly (i.e., from each node of the pair to the node that justified the removal of the link). In order to direct any remaining arrows, common sense graphical rules are applied (see [18] for details).

The conditional independence tests of ICS left out the variable `answertype` from the model as irrelevant. Feature selection (FCBF [22]) also confirmed the relevance of all variables apart from `answertype`. The final model learned appears in Figure 1. To evaluate the result, the `GIC05` dataset (with 590 instances) was employed as a testset (see accuracy report in Figure 2). Further investigation with the data showed that splitting them and considering a different model for every item improves the results substantially (an average of 68.367% accuracy for all items). The main reason behind this, is the fact that some of the variables do not play the same role in every item (for example, the influence of the related items page is not always the same on subsequent items) and therefore, one model cannot accommodate all the items. This process simplified the models considerably and therefore, these separate models were preferred for the actual implementation.

After the implementation of the BN, further investigation with the data revealed that logistic regression is slightly more accurate in certain cases for predicting the need for help (on average it has accuracy 68.92% against the testset and for the model that combines all items 69.89%). Although, as in any model, further investigation and research could improve its accuracy, the model was considered adequate for implementation and further testing. The prediction was employed as a feature in subsequent research related to affective characteristics (see [13]) and plays a particular role as a feature in the machine-learned model described in the next section. The application of the model and how the results could be improved and automated are further discussed at Section 4.

3 Predicting the benefit of students' interactions

Similar to the prediction of unnecessary help-requests, the problem of measuring the benefit of students' interactions in terms of learning is also quite complex and not unique to the research here. Consistent with the choices described in the previous section, the approach taken was to develop a model based on data from all students' interactions with previous applications of the system, correlated with their answers in post-test questions linked to

Table 1. Features considered for learning the model of beneficial interaction

1. Help frequency. 2. Error Frequency
3. Tendency to ask for help rather than risk an error ($\frac{help}{errors+help}$ as defined in [21])
4. No need for help but help requested (according to Section 2) (true/false)
5. Answertype - the type of the answer required (mcq, blank,matrix,checkbox)
6. Previous attempts in items related to the current skill

   If this was the student's first opportunity to practice this skill, -1.

   If no previous attempt was successful, 0.

   Otherwise, a measure of the degree of completeness of the goals of the related item (if there was no related item on the system the standardised score of their exam in the prerequisite of this skill)
7. Time in standard deviations off the mean time taken by all students on the same item.
8. Speed between hints – The Mahalanobis distance of the vector of times between hints from the vector of mean times taken by all students on the same hints and item[3].
9. Accessing the related example while answering (true/false)
10. Self-correction (true/false)
11. Requested solution without attempt to answer (true/false)
12. Reflection on hints (defined as the time until next action from hint delivery) (calculated similarly to 8 using again the Mahalanobis distance).
13. The number of theoretical material lookups that the student followed when such a lookup was suggested by the system (-1 if no lookups were suggested)

specific items in the system. In order to decide the features for the machine learning a combination of qualitative and exploratory analysis was performed (for details see [11]). Based on the results and driven by the predictive power that similar features had in other related research (e.g., [1, 3]) the variables that appear in Table 1 were considered important.

The boolean class learned represents whether a student answered correctly the related post-test question. To achieve a mapping between the actions in the system and the answers in the post-test question the answers are assessed across 4 basic skills that have a direct correspondence with the steps of questions in WaLLiS . In the cases where students did not answer parts of the questions, the missing answers are considered as wrong (i.e. the boolean variable is FALSE). In addition, in average 8 students in every dataset did not interact with certain steps or whole items in WaLLiS and therefore their data were excluded. In fact, most of these 26 students did not attempt to answer the post-test question and the few who did provided wrong answers. This resulted in 472 instances from `GIC04`, 352 from `GIC05`.

For similar reasons as the ones described in the previous section, BNs were preferred. Informal comparisons with decision trees established that they had similar accuracy. Whilst

[3]The Mahalanobis distance is used in the place of the traditional Euclidean distance. It utilises group means and variances for each variable as well as the correlations and covariance of the data set. It is usually used as a metric to test whether a particular instance would be considered an outlier relative to a set of group data. Formally, the distance of a vector $x = (x_1, x_2, \ldots, x_n)^T$ from another vector, $y = (y_1, y_2, \ldots, y_n)^T$ is defined as $D(x) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}$ where Σ is the covariance matrix.

Table 2. Variables after feature selection from Table 1

1.	Tendency for help
2.	Need for help
3.	Self-correction
4.	Example access
5.	Average reflection time
6.	Speed for hints
7.	Error frequency

Table 3. Classification accuracy and Kappa statistic for Bayesian networks and tree induction to predict beneficial interaction

		BayesNet		J4.8	
		Cross	Test Set	Cross	Test
	accuracy	70.112	68.234	66.517	65.843
	Kappa	0.401	0.364	0.318	0.313
recall	True	0.726	0.726	0.714	0.686
recall	False	0.672	0.623	0.605	0.626

implementing a model based on decision trees would have increased the flexibility when providing feedback, it is not clear if communicating to the students the reasons behind the ineffective interaction, would have any effect, particularly if one takes into account that in more than 30% of the cases the system could be wrong. In addition, in future implementations, this prediction could be used as part of the evidence for a holistic framework.

To learn the BN the ICS algorithm of Weka was employed again and to facilitate the algorithm's search feature selection is also employed in advance to remove irrelevant and redundant features. Although (as expected) the same more or less accuracies and structure was learned over the complete set of features, simpler models are always preferred. In fact, the simplified model achieved better accuracy on a 10-fold cross validation check and slightly better accuracy on the test set. By removing redundant features the remaining ones were easier to comprehend. This allows a more sensible ordering of the variables, which can effect the search for the structure of the ICS algorithm. Finally, the process with feature selection was significantly faster. In this case, since all the analysis was performed offline, and the models were implemented prior to their integration with the system, this is not really relevant. However, if the techniques reported here are automated to enable the system to learn while more students work with it, online speed will become a more critical factor. The list of reduced variables is shown in Table 2 and the final model in Figure 3. Its accuracy report, as well as comparisons with decision trees are presented in Table 3.

Logistic regression was tested again after the implementation of the BN. It provides an accuracy of 70.34% against the testset and although it not a significant improvement it is definitely worth considering in the future. The next section describes how the two models described were employed when redesigning WaLLiS and future work for improving them.

4 Application and Future Work

As discussed in the Introduction the necessity behind the development of the two models was established through qualitative research aiming to improve WaLLiS by taking into account students' affective characteristics. In particular, as presented in [13], the two models are interrelated as the prediction of the necessity of help requests appears in rules related to affective factors (particularly effort) but also the model of beneficial interaction seems to have the potential to model tutors' decisions and feedback.

The immediate benefit of the two models was to empower the system with a measure-

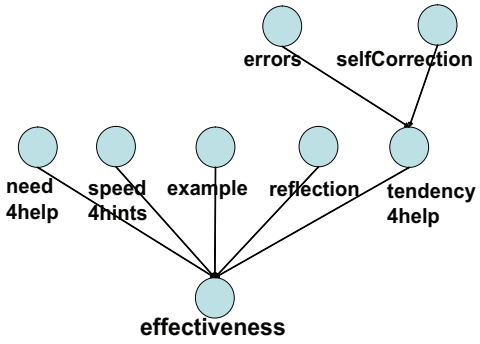


Figure 3. Bayesian Network for predicting beneficial interaction

ment of desirable interaction on the basis of which feedback, interventions and suggestions for studying further material can be provided. The two models were implemented (using JavaBayes[4]) as part of a diagnostic agent the outcomes of which are taken into account by the system's feedback mechanism in order to adapt its actions. Accordingly, when students are asking for suggestions on what to study next, the prediction of beneficial interaction is employed assisting the system to prioritise the available items for the student. In other cases students may have just completed an item but because of the way they interacted with it, the model predicts a low benefit of their interaction. The system then suggests they try the exercise again. The prediction is also employed when students are asking for suggestions within an item rather than a specific hint on a step. If the prediction is high, a positive comment is provided followed by an encouragement to complete the remaining items (if any) before moving on to the next activity. Otherwise, they are reminded of the goals of the exercise they are interacting with and are encouraged to complete the current item first. When requesting specific hints in steps, the prediction for the necessity of help-requests is used but it does not play a direct role. In other words, it does not prevent students for asking help but it plays an indirect role in the model of beneficial interaction.

Future work will focus into improving the models. The separation of the models by items seems to be against the long-term goal of coming up with one model that could be used in other lessons. On the other hand, the methodology for building it can be used in other contexts especially if it is automated allowing the system to learn and improve itself while it is used rather than when offline. Also, it was discussed that logistic regression seems to provide better accuracy and therefore future work will investigate its implementation. Logistic regression is not only more straightforward to implement but it can also provide the probabilistic framework needed to deal with the uncertain nature of the predictions. In addition, it is worth observing that the prediction from the model of beneficial interaction can in its turn be used as a feature in the model for the necessity of help requests in the place of the *rel* variable thus avoiding the use of the subjective rule-based measurement and relying to the more valid model learned from data.

The development of the models introduced a qualitatively different interaction. Therefore, future work will focus on evaluating the different decisions and the impact they may

[4]http://www.cs.cmu.edu/ javabayes/

have in students' learning. The design choices for the feedback mechanism are already influenced by the accuracy and the probabilistic nature of the prediction. As also discussed in [1, 3], The approach taken should not be too intrusive (e.g., interrupting students' work in order to provide feedback) nor preventative (e.g., preventing them from asking help). In particular, if one takes into account that in around 30% of the cases the model could be wrong it is obvious, but paramount, to use these predictions in a way that has the least negative educational consequences. The question is how to strike a balance between an approach that utilises the predictions from the models in an informative way and a more preventative approach that may be required in some cases. In the cases where communicating this information with the student becomes problematic, models such as the ones developed here could have other applications. For example, the prediction of the benefit of the interaction could be used in open-learner modelling or in classroom environments to provide useful information for teachers on the basis of which they can act.

Finally, it is worth considering in more detail the evaluation of the models, since they influence the judgements on the validity of the results. In the case of the models presented here, an average of 70% was considered satisfactory because of the design choice in relation to not following a preventative approach. A suggestion of repeating an exercise, even when given under false evidence, is probably not that problematic compared to not allowing students to request help just because the system thinks (sometimes wrongly) that their request is superfluous. Therefore, in Educational Data Mining, it is worth taking into account the nature of the data and the 'cost' of true or false positives rather than evaluating them out of context. Given the availability of data that include students' performance, future work will focus on taking more informed decisions about the models and the appropriate actions of the system in terms of being detrimental to students' learning.

Acknowledgments The work presented here is part of the author's PhD research, partially funded by The University of Edinburgh. The author would like to thank Antony Maciocia for the general support throughout this research and in particular for the access to the GIC classrooms that made the data collection possible. Also, Ryan Baker and the members of the WEKA mailing list for discussions related to the data mining tools employed.

Bibliography

[1] Aleven, V. and Koedinger, K. (2000). Limitations of student control: Do students know when they need help? In *ITS*, volume 1839 of *LNCS*, pages 292–303.

[2] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[3] R.S. Baker, A.T. Corbett, K.R. Koedinger, and A.Z. Wagner. Off-task behavior in the cognitive tutor classroom: When students "game the system". In *Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems, 24-29 April 2004, Vienna, Austria*, pages 383–390. ACM Press, New York, 2004.

[4] R.R. Bouckaert. Bayesian networks in Weka. Technical report, Computer Science Department University of Waikato, 2004.

[5] C. Conati, A. Gertner, K. VanLehn, and M. Druzdzel. On-line student modeling for coached problem solving using bayesian networks. In A. Jameson, C. Paris, and C. Tasso, eds, *User Modeling: Proceedings of the 6th International conference*, 1997.

[6] A.T. Corbett and J.R Anderson. Student modeling and mastery learning in a computer-based programming tutor. *ITS*, volume 608 of *LNCS*, pages 413-420, 1992.

[7] M R. Lepper, M. Woolverton, D L. Mumme, and J. Gurtner. Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S.P. Lajoie and S.J. Derry, eds, *Computers as Cognitive Tools*, pages 75–107. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1993.

[8] J. Lin, E. Keogh, S Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. *Data Mining and Knowledge Discovery*, 2003.

[9] J. Martin and K. Vanlehn. Student assessment using bayesian nets. *International Journal of Human-Computer Studies*, 42:575-591, 1995.

[10] M. Mavrikis. Logging, replaying and analysing students' interactions in a web-based ILE to improve student modelling. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, page 967, Amsterdam, 2005.

[11] M. Mavrikis. *Modelling Students' Behaviour and Affective States in ILEs through Educational Data Mining*. PhD thesis, The University of Edinburgh, 2008.

[12] M. Mavrikis and A. Maciocia. WaLLiS: a web-based ILE for science and engineering students studying mathematics. *Supplemental Proceedings of the 11th International Conference on Artificial Intelligence in Education, AIED2003, Vol. VIII, Workshop on Advanced Technologies for Mathematics Education*, pages 505–513, 2003.

[13] M. Mavrikis, A. Maciocia, and J. Lee. Towards predictive modelling of student affect from web-based interactions. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, pages 169–176, Los Angeles, California, 2007.

[14] R.C. Murray and K. VanLehn. Effects of dissuading unnecessary help requests while providing proactive help. In *Proceedings of the 12th International Conference on Artificial Intelligence In Education*, pages 887–889., Amsterdam, 2005.

[15] R.S. Newman. Students' help seeking during problem solving: Influences of personal and contextual achievement goals. *Journal of Educational Psychology*, 90:644–658, 1998.

[16] K. Porayska-Pomsta, M. Mavrikis, and H. Pain. Diagnosing and acting on student affect: the tutor's perspective. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, 18(1-2):125–173, 2008.

[17] K. VanLehn and J. Martin. Evaluation of an assessment system based on bayesian student modeling. *Int. Journal of AI in Education*, 42:575-592, 1998.

[18] T. Verma and J. Pearl. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 323–330, 1992.

[19] B. Weiner. *Human motivation: Metaphors, theories, and research*. Sage, 1992.

[20] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.

[21] H. Wood and D. Wood. Help seeking, learning and contigent tutoring. *Computers & Education*, 33:153–169, 1999.

[22] L. Yu and L. Huan. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *International Conference on Machine Learning*, 2003.

Integrating Knowledge Gained From Data Mining With Pedagogical Knowledge

Roland Hübscher[1] and Sadhana Puntambekar[2]
rhubscher@bentley.edu, puntambekar@education.wisc.edu
[1] Department of Information Design and Corporate Communication, Bentley College
[2] Department of Educational Psychology, University of Wisconsin

Abstract. Discovering knowledge from raw data is one of the goals of data mining. Yet, it is not always clear how this knowledge is used in educational computing systems and how exactly it is integrated with other knowledge like the pedagogy used. We present a case study where the use of the data mining results was initially described, to a large degree, at the implementation level, thus largely ignoring the nature of the different kinds of knowledge involved. Based on Clancey's heuristic classification model [7], the description is raised to a conceptual level, the knowledge level. This results in an explicit and well-defined integration of knowledge discovered with data mining techniques, pedagogical knowledge and linguistic knowledge. Such a knowledge-level description leads to an improved understanding of the system and its effects on the learners.

1 Introduction

Educational data are mined with the goal to discover knowledge about the learners, educational software and other classroom interventions. Thus, the designers need to be explicit about how that knowledge is being used to redesign educational software. Yet, many of us working in the general area of educational technology too often talk about software or more general interventions at the implementational level. Staying at that level leaves the use of the data mining knowledge and its integration with pedagogical knowledge implicit.

Having struggled with this issue ourselves, we present in this paper a case study from our own research. CoMPASS is an educational hypertext system helping middle-school students learn science. Originally, we had used data mining to better understand the design of CoMPASS, most notably the impact of a concept map as navigation support [16, 18]. Analysis using data mining techniques showed that certain navigation patterns are indicative of student learning. We then decided to take advantage of these patterns and use them to provide the students with adaptive and context sensitive prompts which are consistent with the notion of scaffolding [17]. Prompting a student with a question or a statement is a tool frequently used by teachers to make the student reflect on issues that will hopefully result in an improved understanding or problem solving behavior. Based on the user categorization found with data mining, we developed a simple scheme to generate such prompts [16]. However, the mechanism for generating the prompt was quite ad hoc and, with hindsight, it was not clear enough what roles pedagogy and insights gained from data

mining played. This was one of the main reason its implementation stalled. Therefore, we developed a framework for our prompt-generating scheme addressing these issues.

It must always be clear what knowledge led to which design decisions. For instance, was there a valid pedagogical reason to make the user select between two choices, or was it computationally infeasible to reduce the "choice" to one item? Thus, design decisions and their reasons have to be made explicit. Designers of adaptive hypermedia systems [4], for instance, have dealt with this issue. Should an educational adaptive hypermedia system have an explicit pedagogical model describing pedagogical knowledge, or is it fine to fold the pedagogical assumptions into the adaptive engine itself? Quite a lively discussion around this topic happened in an after-session meeting at Adaptive Hypermedia 2004. The preferred, though surprisingly not unanimous, view was that indeed, the pedagogical model should be separate. Another example is the confusion between the concept of prerequisite and pedagogy [11]. The fact that a concept A is a prerequisite for understanding a concept B does not imply that A must be taught before B as some adaptive hypermedia systems assume [5]. Again, different kinds of knowledge are mixed up which leads to important implicit design decisions that should have been made explicit.

In this paper, we will present the case study as follows. First, we will introduce the educational hypertext system CoMPASS and the design of scaffolding with adaptive prompts. Therefore, scaffolding in educational systems will be briefly described before presenting the original design described at the implementational level. We then show how this description can be raised to the conceptual knowledge level making explicit the role the different kinds of knowledge play and how they can be integrated. Although this is a specific case study, it will become quite apparent how other designs suffering from similar problems can take advantage of the approach described here.

2 Scaffolding in CoMPASS with Prompts

2.1 CoMPASS

CoMPASS, shown in Figure 1, is a hypertext system to help middle-school students learn science. CoMPASS presents students with two representations, a textual representation of concept descriptions and a graphical representation in the form of a navigable concept map. Both representations change dynamically as students traverse through the domain and make reading choices.

Each page in CoMPASS is a description of a concept within a certain topic. In a domain like physics such concepts would include force, mass, acceleration, etc. When students choose a concept, CoMPASS presents them with a description of that concept along with a navigable map that shows them the related concepts. Instead of presenting a hierarchically organized set of topics with the option of going to a glossary page for related information, the very basis for navigation in CoMPASS is associative relationships between conceptual units. The maps provide local coherence, by showing students the concepts that are most related to the concept that they have chosen. In CoMPASS, the navigable maps are drawn with a fisheye, with the concept that the student has chosen as the focus. The rest of the map is drawn by retrieving the most related concepts from a database, which stores the semantic relatedness information of the concepts. We have used the relationship strength to

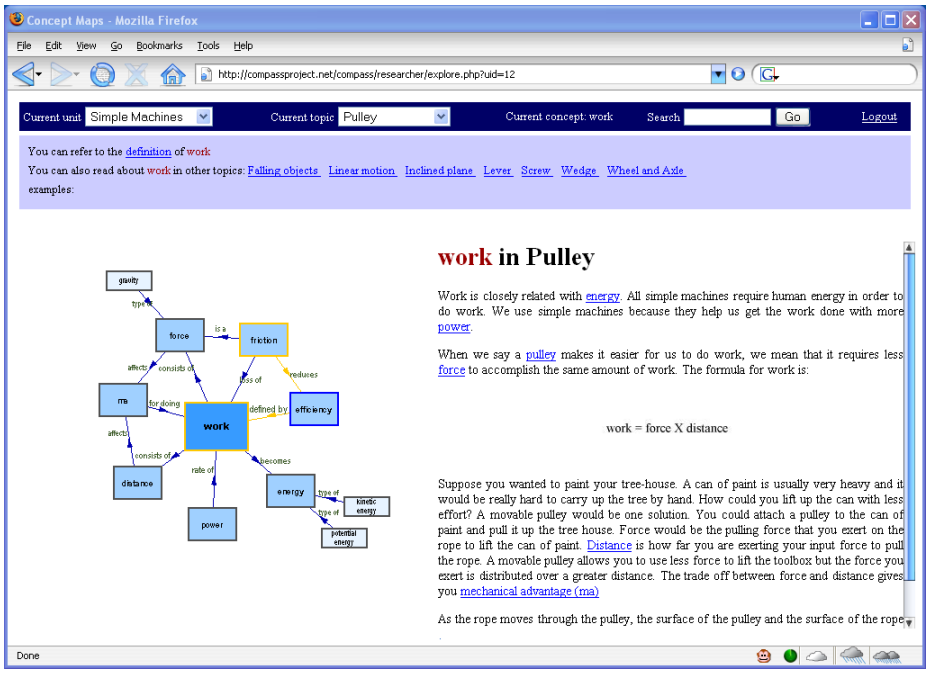


Figure 1. CoMPASS with the concept map supporting navigation on the left and the content (concept 'work' as it is used in the context of the simple machine' pulley') on the right.

determine the spatial proximity of the concepts. Thus the stronger the relationship between the two concepts, the closer they are spatially in the concept map.

In CoMPASS, students can easily switch views to go to a related topic. This provides global coherence, because students can see what other topics they can go to that could be related to a particular topic. In addition, they can also view a particular concept from multiple perspectives as described below. For example, a student setting up an experiment with a pulley might be interested in learning about 'work' in the context of a lever instead of pulley as shown in Figure 1. Thus, the student can navigate within a context (e.g., pulley) or across (e.g., from pulley to lever). Learning in a subject area, such as science, involves understanding the rich set of relationships among important concepts, which may form a web or a network. Revisiting the same material at different times, in rearranged contexts, for different purposes, and from different conceptual perspectives is essential for attaining the goals of advanced knowledge acquisition [22]. The alternative views that CoMPASS offers can help students to study science concepts and phenomena in depth by visiting them in multiple contexts.

3 Scaffolding with Prompts

A next step is to add adaptive support with textual prompts that help students directly, especially when they have some problems, and indirectly also teachers who cannot attend to all students in the classroom at all times. The prompts are supposed to *scaffold* the students. Since the concept of scaffolding has been somewhat overused [17], we briefly describe it.

Scaffolding in the context of learning has originally been defined as an "adult controlling those elements of the task that are essentially beyond the learner's capacity, thus permitting him to concentrate upon and complete only those elements that are within his range of competence" [27]. Scaffolding has been linked to the work of Soviet psychologist Lev Vygotsky, although he never used the term scaffolding. According to Vygotsky, a novice learns with an expert, and learning occurs within the novice's Zone of Proximal Development (ZPD). ZPD is defined as the "distance between the child's actual developmental level as determined by independent problem solving and the higher level of potential development as determined through problem solving under adult guidance and in collaboration with more capable peers" [25]. Enabling the learner to bridge this gap between the actual and the potential depends on the resources or the kind of support that is provided. Instruction in the ZPD can therefore be viewed as taking the form of providing assistance or scaffolding, enabling a child or a novice to solve a problem, carry out a task or achieve a goal "which would be beyond his unassisted efforts" [27].

Proper scaffolding requires a computer-based learning environment like CoMPASS to support, among other things, (a) continuous assessment of the learner needs to be used to calibrate the support; (b) scaffolding fading away over time and the learner taking control of the task; and (c), the learner needing to be actively involved in the learning process [15,23].

The implications for the prompts are therefore: (a) the prompts must be adapted to the student's current understanding and progress, i.e., they must be adaptive, context sensitive and individualized; (b) the prompts should be formulated and presented (or not!) so that the student is not "bothered" by them when there is no need for support anymore; and (c), the prompts should be formulated such that they result in active reflection and they are not just corrective suggestions to be followed mindlessly.

Adaptive support requires modeling users as in, for instance, adaptive hypermedia systems where mostly explicit user models are used by the system to adapt presentation and navigation support to each individual user [4]. However, this is simply not feasible given how CoMPASS is being used. Only sparse user data is available and there is no time to collect detailed user information with questionnaires or multiple-choice tests. We basically have to rely on a few clicks to detect how a student is progressing.

Fortunately, earlier work on data mining the navigation data collected from the CoMPASS users had revealed that students using CoMPASS can be assigned to categories that can be associated with the students' approach to learning and understanding of the material [19].

Since this paper's focus is on integration of the discovered knowledge and not on the discovery process itself, we just briefly summarize the data mining methods used. To find the learner categories, each student's clickstream was converted into a navigation matrix N describing the number of transitions $N_{i,j}$ from concept c_i to concept c_j and then pruned using the Pathfinder algorithm [21]. The resulting matrices, one for each user, were then clustered using the k-Means algorithm [10]. The students in these clusters were then analyzed to see what educational characteristics they had in common. For instance, as described in [19], the students in one cluster showed that they focused on the relevant (as determined by an expert) concepts but also visited related concepts. Such students tended to do well. In another cluster, students apparently had no well-defined focus and explored concepts also in other topics. Yet another group showed a random behavior indicating that

Condition: The ratio of visited to existing related topics for a concept *C* is small.
Justification: The student does not visit a certain concept *C* in related topics. Encourage student to read the same concepts in related topics.
Example prompt: Have you read concept *C* in any other topic?

Figure 2. A rule described at the implementational level.

they were not aware of the conceptual structure of the domain.

Based on these results, we developed rules to generate adaptive prompts [16]. These rules use some simple characteristics of the realtime navigation data, i.e., the clickstream, to detect behavior associated with the clusters found during data mining. An example of such a rule is shown in Figure 2. Of course, this is not the implementation of the rule itself, but it is described in terms of concepts at the level of the implementation, especially the condition. The justification is not used by the system and only serves as a comment to the writer of the rule.

This approach sounds reasonable and is described in more general form in Figure 3. However, as the figure shows, the role that the pedagogical knowledge plays and how it connected to the classifications generated by the data mining methods is conceptually not very clear. The reason is that the approach is described at the implementational level instead of the knowledge level. Thus, modifications to the data mining approach, to the pedagogical approach and to what kinds of text prompts to use will have to happen at the implementational level.

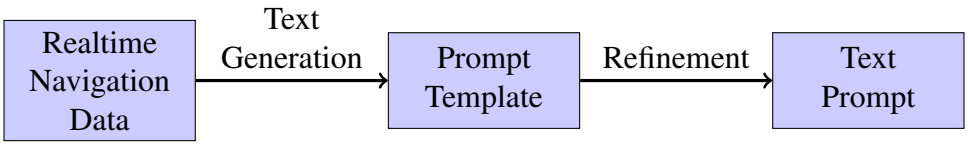


Figure 3. Text prompt generation algorithm.

4 Integrating Data Mining with Pedagogical Knowledge

One of the pitfalls of system design is to make decisions at the implementational level when they should be made at the conceptual level, although, of course, implementational constraints need to be considered. For instance, in the early AI days, expert system developers argued about forward versus backward chaining, instead of focussing on what tasks the experts solved and with what problem solving methods. Fortunately, the discussion soon moved from the implementational level to the conceptual, or, knowledge level [26]. This type of work benefited from earlier research by Clancey on classification of the reasoning that goes on in expert systems [7].

Clancey [7] describes the simple "heuristic classification" inference structure in Figure 4 that provides the basis for many problem solving methods used by experts. For instance, from a set of symptoms (data) a doctor abstracts to a class of symptoms (data

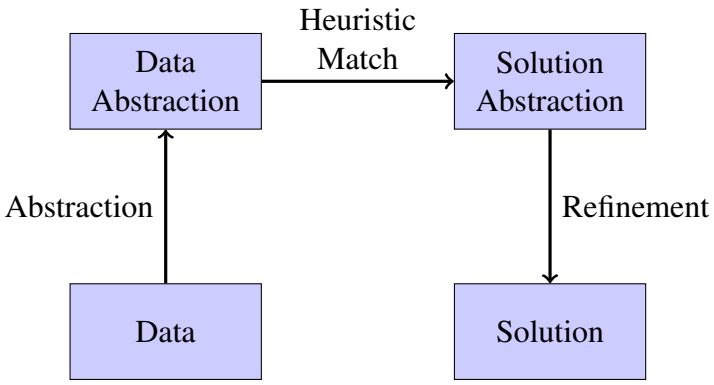


Figure 4. Heuristic classification.

abstraction) which then requires a certain type of treatment (solution abstraction) which is found by applying medical knowledge (heuristic match). Using contextual information about the patient and treatment, the type of treatment can be refined, e.g., the dosage can be adjusted (refinement).

More direct and simpler symptom-to-treatment reasoning is not as good, since it does not take advantage of the different types of knowledge (classification, refinement, heuristic medical knowledge) and it would result in a much less effective representation of the knowledge. Adding new knowledge would be messy since it would not be clear how exactly it should be integrated and used. Just reacting to symptoms before one is able to classify the type of a problem would also result in more mistreatment. If new medications or symptom detectors are introduced, it is clear how they are going to be integrated in the heuristic classification scheme.

Figure 5 shows the heuristic classification scheme applied to the generation of adaptive prompts. The direct link from the realtime navigation data to the text prompts (see Figure 3) has been replaced by a conceptual structure integrating the various types of knowledge involved. It should not be a surprise that the heuristic classification scheme is such a good match. After all, the learner's navigation behavior is the symptom and the text prompt the treatment for the learner.

In Figure 5, there are three types of knowledge explicitly represented. The knowledge discovered with the data mining methods is represented by the learner categories. Pedagogical knowledge is used to decide what kind of textual interventions should be used. And finally, linguistic knowledge is used to create the appropriate text. The latter is especially important if several interventions need to be combined into one phrase, or past prompts need to be taken into account.

Applying the problem solving method in Figure 5 results in the rule set shown in Figure 6. There are now three steps instead of one. First, the student's navigation behavior is still categorized the same way as in the original version, but this time mapped explicitly to one or more of the learner categories discovered via data mining of the off-line data collected in the past. Since the categories are the result of the data mining process, once improved data mining results are available—and we are working on them—these categories

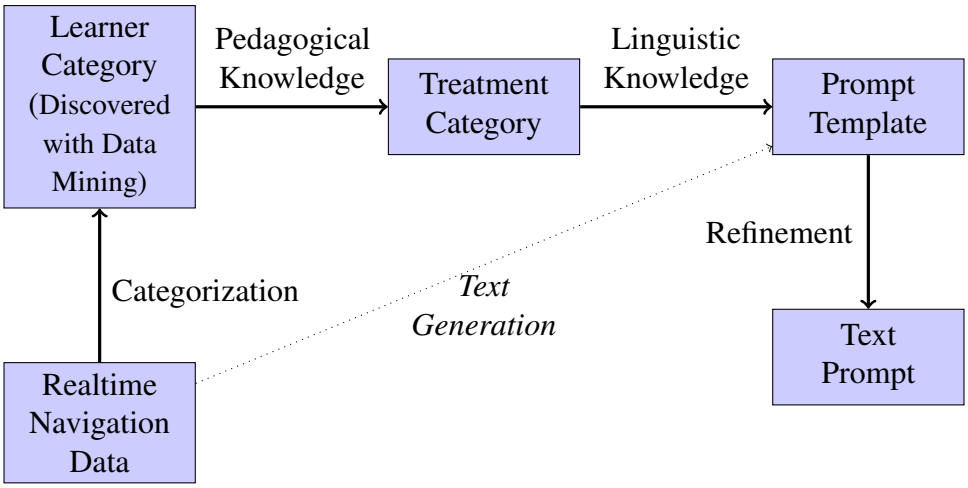


Figure 5. Text prompt generation at the conceptual level using the heuristic classification scheme. The dotted line refers to Figure 3.

can be modified.

Second, based on the learner categories and the used pedagogy, the kind of feedback deemed most useful for such a learner is suggested. Although we intend to use as simple rules as shown in Figure 6, a much more complicated reasoning process could be involved, however, using pedagogical knowledge only. Third, the treatments are collected and translated into a proper prompt. For instance, if the first rule (categorization) in Figure 6 applies to two concepts, the second linguistic rule will be applied. We will use a relatively simple yet quite powerful template-based approach to generate the natural language output [8, 20]. This will also allow us to take previous prompts into consideration and avoid repeating the same prompt over and over even if the student does not improve. In our approach, Clancey's heuristic match (see Figure 4) is composed of two steps. The refinement step is the same as in the old case where the necessary variables in the template are bound based on the context.

In the original formulation of how the prompts were generated, all three kinds of knowledge were mixed into each rule. Theoretically as well as practically, this is a problem. From a theoretical point of view, it is quite unclear what types of knowledge are involved and how. For instance, the importance of the linguistic knowledge was originally overlooked. From a practical point of view, the modular approach makes it clear what to change, be the change either due to modification in the pedagogical approach or due to improved data mining or text generation methods.

This approach is not only suitable for the specific situation in CoMPASS. An obvious place to apply the approach described in this paper is in the context of association rules [9, 12]. These rules capture associations like *beer* $\Rightarrow$ *diapers* between variables of items in shopping baskets [2]. Based on our experience in CoMPASS, it will be useful to always carefully consider and make explicit what type of knowledge such rules capture. Looking at the current literature, it seems, that this is generally not done in an explicit and rigorous way. Such association rules are normally judged by some mathematical definition

Categorization
Condition: The ratio of visited to existing related topics for a concept *C* is small
Action: Assign student to category `ignores_related_topics`

Pedagogical knowledge (heuristic match)
Condition: Student `ignores_related_topics`
Action: Say something to encourage reading concept *C* in other topics.

Linguistic knowledge (heuristic match)
Condition: Encourage reading concept *C* in other topics
Action: Create prompt "Have you read concept *C* in any other topic?"

Condition: Encourage reading concept C_1 in other topics *and* encourage reading concept C_2 in other topics
Action: Create prompt "Have you read concepts C_1 and C_2 in any other topic?"

Figure 6. A set of rules based on the conceptual description shown in Figure 5.

of interestingness [1, 3, 14] which is perfectly fine for finding the rules. However, the rules also need to be used in a meaningful way and that is where our approach may be useful.

5 Conclusions

We have presented a case study of our own research showing that making the different kinds of knowledge including the ones gained from data mining can have great benefits. It is clear what modifications need to be made if pedagogy, data mining techniques, or other parts are being changed. This is especially useful if weak points need to found in the educational software. It may not be *what* the prompts say, but *how* they say it. This is similar to the difficulty students have with word problems where often difficulties reflect students' language problems but not necessarily their math problems [24].

Although this case study presented a specific problem and solution, the idea of making the problem solving and the knowledge involved explicit, is a general one and should always be done be that based on Heuristic Classification [7], KADS [26], Generic Tasks [6] or another method to model knowledge and problem solving methods [13]. We have used the heuristic classification because it is simple, yet powerful enough.

But is this approach more than just proper system design? Well, not really. However, it does matter at what level we design, experiment with, modify and understand a system, and take advantage of various kinds of knowledge. This is crucial if we work on complex systems supporting learners where it surely is not good enough if "it works." We also must understand *why* it works.

Acknowledgments

This material is based upon work supported in part by the NSF under Grants #0437660 (IERI) and #0434624 (NSDL). Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] R. Agrawal, T. Imielinski, and A. A. Swami. Mining associations between sets of items in large databases. In *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pages 207–216,, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, 1994.

[3] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, 1997.

[4] P. Brusilovsky. *Methods and Techniques of Adaptive Hypermedia*, volume 6, pages 87–129. Kluwer Academic Publishers, 1996.

[5] P. Brusilovsky, E. Schwarz, and G. Weber. ELM-ART: An intelligent tutoring system on world wide web. In C. Frasson, G. Gauthier, and A. Lesgold, editors, *Intelligent Tutoring Systems (Lecture Notes in Computer Science)*, volume 1086, pages 261–269. Springer Verlag, Berlin, 1996.

[6] B. Chandrasekaran, M. C. Tanner, and J. R. Josephson. Explaining control strategies in problem solving. *Expert Systems in Government*, pages 9–24, 1989.

[7] W. J. Clancey. Heuristic classification. *Artif. Intell.*, 27(3):289–350, 1985.

[8] K. V. Deemter, E. Krahmer, and M. Theune. Real versus template-based natural language generation: A false opposition? *Comput. Linguist.*, 31(1):15–24, 2005.

[9] C. García, Enrique Romero, S. Ventura, and T. Calders. Drawbacks and solutions of applying association rule mining in learning management systems. In *Proceedings of the International Workshop on Applying Data Mining in e-Learning (ADML'07)*, pages 14–22, 2007.

[10] P. Hansen and N. Mladenovic. J-Means: A new local search heuristic for minimum sum-of-squares clustering. *Pattern Recognition*, 34(2):405–413, 2001.

[11] R. Hübscher. What's in a prerequisite. In *Proceddings of the International Conference on Advanced Learning Technology (ICALT 2001)*, pages 165–168, Madison, 2001.

[12] A. Merceron and K. Yacef. Revisiting interestingness of strong symmetric association rules in educational data. In *Proceedings of the International Workshop on Applying Data Mining in e-Learning (ADML'07)*, pages 3–12, 2007.

[13] E. Motta. *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1999.

[14] E. R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, 2003.

[15] A. S. Palincsar. Keeping the metaphor of scaffolding fresh–a response to c. addison stone's "the metaphor of scaffolding: Its utility for the field of learning disabilities". *Journal of Learning Disabilities*, 31(4):370–73, 1998.

[16] S. Puntambekar. Analayzing navigation data to design adaptive navigation support in hypertext. In U. Hoppe, F. Verdejo, and J. Kay, editors, *Artificial Intelligence in Education: Shaping the future of learning through intelligent technologies*, pages 209–216. IOS Press, 2003.

[17] S. Puntambekar and R. Hübscher. Tools for scaffolding students in a complex learning environment: What have we gained and what have we missed? *Educational Psychologist*, 40(1):1–12, 2005.

[18] S. Puntambekar and A. Stylianou. Designing metacognitive support for learning from hypertext: What factors come into play? In U. Hoppe, F. Verdejo, and J. Kay, editors, *Artificial Intelligence in Education: Shaping the future of learning through intelligent technologies, workshop proceedings*. IOS Press, Amsterdam, 2003.

[19] S. Puntambekar, A. Stylianou, and R. Hübscher. Improving navigation and learning in hypertext environments with navigable concept maps. *Human-Computer Interaction*, 18(4):395–428, 2003.

[20] E. Reiter and R. Dale. *Building Natural Language Generation Systems.* Cambridge University Press, 2000.

[21] R. W. Schvaneveldt, editor. *Pathfinder Associative Networks: Studies in Knowledge Organization*. Ablex, Norwood, NJ, 1990.

[22] R. J. Spiro, P. J. Feltovich, M. J. Jacobson, and R. L. Coulson. Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. *Educational Technology*, May:24–33, 1991.

[23] C. A. Stone. The metaphor of scaffolding: Its utility for the field of learning disabilities. *Journal of Learning Disabilities*, 31(4):344–364, 1998.

[24] T. A. Van Dijk and W. Kintsch. *Strategies of discourse comprehension*. New York: Academic, 1983.

[25] L. S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, MA, 1978.

[26] B. J. Wielinga, A. T. Schreiber, and J. A. Breuker. Kads: a modelling approach to knowledge engineering. *Knowl. Acquis.*, 4(1):5–53, 1992.

[27] D. Wood, J. S. Bruner, and G. Ross. The role of tutoring in problem solving. *Journal of Child Psychology & Psychiatry & Allied Disciplines*, 17(2):89–100, 1976.

Can an Intelligent Tutoring System Predict Math Proficiency as Well as a Standardized Test?

Mingyu Feng[1], Joseph Beck[1], Neil Heffernan[1], and Kenneth Koedinger[2]
[1] Department of Computer Science, Worcester Polytechnic Institute
[2] Human Computer Interaction Institute, Carnegie Mellon University

Abstract. It has been reported in previous work that students' online tutoring data collected from intelligent tutoring systems can be used to build models to predict actual state test scores. In this paper, we replicated a previous study to model students' math proficiency by taking into consideration students' response data during the tutoring session and their help-seeking behavior. To extend our previous work, we propose a new method of using students test scores from multiple years (referred to as cross-year data) for determining whether a student model is as good as the standardized test to which it is compared at estimating student math proficiency. We show that our model can do as well as a standardized test. We show that what we assess has prediction ability two years later. We stress that the contribution of the paper is the methodology of using student cross-year state test score to evaluate a student model against a standardized test.

1 Introduction

In many states there are concerns about poor student performance on new high-stakes standards-based tests that are required by No Child Left Behind (NCLB). For instance, the Massachusetts Comprehensive Assessment System (MCAS) administers rigorous standardized tests in English, math, history and science in grades 3–12. Students need to pass the math and English portions of the 10th grade versions in order to get a high school diploma without further remediation. In 2003, a full 10% of high school seniors were predicted to be denied a high school diploma due to having failed to pass the test on their fourth try. Moreover, the state of Massachusetts has singled out student performance on the 8th grade math test as an area of highest need for improvement[1]. Partly in response to this pressure, and partly because teachers, parents, and other stakeholders want and need more immediate feedback about how students are doing, there has recently been intense interest in predicting student performance on end-of-year tests [12]. Some teachers make extensive use of practice tests and released test items to help identify learning deficits for individual students and the class as a whole. However, such formative assessments not only require great effort and dedication, but they also take valuable time away from instruction. Some online testing systems (such as Renaissance Learning[2]) automatically grade students and provide reports but they may not be informative as they do not maintain sufficiently rich data records for students and therefore cannot report on a fine-grained model of student knowledge.

[1] http://www.doe.mass.edu/mcas/2002/results/summary.pdf
[2] www.renlearn.com

The ASSISTment system (http://www.assistment.org) is an attempt to blend the positive features of both computer-based tutoring and benchmark testing. It helps students to work through tough problem by breaking the problem into steps; meanwhile, it collects data related to different aspects of student performance such as accuracy, speed, help-seeking behavior and attempts as students interact with the system. Based on the rich source of data, various reports (e.g. [10]), for individual students and for groups, have been developed to help teachers and other stakeholders to better understand students' performance and progress. Furthermore, we have done research (e.g. [1], [2], [8], [9]) to train models that use the ASSISTment metrics to accurately predict state test scores.

In this paper, we replicated the study in [8] to model students' math proficiency for Spring 2005 by taking into consideration students' response data during the tutoring session and their help-seeking behavior. Among these studies on modeling student proficiency, different criteria have been used to evaluate the effectiveness of the student models. For instance, Beck & Sison [3] used relative closeness to real scores and mean absolute error (MAE); Anozie & Junker ([1]) evaluated their model using mean absolute deviation (MAD) which is essentially the same as MAE. Feng, Heffernan & Koedinger ([8]) reported R square and Bayesian Information Criterion (BIC) of the models we fitted. In this paper, we propose a new method of using students' cross-year test scores (from the year 2007) to evaluate the effectiveness of the student model on predicting student math proficiency. We will compare our prediction with the 2005 MCAS test on how well each correlated with the 2007 MCAS scores, assuming student math proficiency at 8^{th} grade and at 10^{th} grade are highly correlated with each other. The paper will be organized as follows. We will briefly describe the ASSISTment system in section 2 and then in section 3 present our approach, including the data we used for this study, the modeling process and the results. The analysis of the results will be discussed in the fourth section. Finally, we conclude in section 5 and bring up some future work.

2 ASSISTment - The Intelligent Tutoring System

Traditionally, the areas of testing (i.e. Psychometrics) and instruction (i.e., math educational research and instructional technology research) have been separate fields of research with their own goals. To address this issue, we built a web based e-learning system that integrates assistance and assessment. The system, named ASSISTment (e.g. [14], [15]), was built to offer instruction to students while providing a more detailed evaluation of their abilities to the teacher than is possible under current approaches. The key feature of ASSISTments is that they provide instructional support (i.e. assistance) in the process of assessing students. In Massachusetts, the state department of education has released ten years (1998-2007) worth of MCAS test items, over 300 items, which we have turned into ASSISTments by adding "tutoring." If students get an item (called the *original question*) correct they will be given a new item. If they get it wrong, they are provided with a small "tutoring" session where they are forced to answer a few (3 ~ 5) questions (called *scaffolding questions*) that break the problem down into steps to eventually get the problem correct. Students are allowed to ask for hints when they encounter difficulty answering a question. Although the system is web-based and hence accessible in principle anywhere/anytime, students typically interact with the system during one class period in the schools' computer labs every three or four weeks. As

students interact with the system, the background logging system collects detailed data of student responses. The hypothesis is that ASSISTments can do a better job of assessing student knowledge limitations than practice tests or other online testing approaches by using a "dynamic assessment" approach based on the data collected online. In particular, ASSISTments use the amount and nature of the assistance that students receive as a way to judge the extent of student knowledge limitations.

3 Approach

3.1 Data Source

The data we consider comes from the 2004 – 2005 school year, the first full year in which the ASSISTment system was used in classes in two middle schools in Massachusetts. The data set contains online interaction data of 417 8th grade students from the ASSISTment system, the results of 8th grade MCAS tests taken in May, 2005 and the results of 10th grade MCAS tests taken by the same group of students two years later, in May, 2007. Since the purpose of this study is comparing the prediction power of ASSISTment with MCAS and there are 39 questions in the MCAS tests, we excluded the data of the 25 students who have done fewer than 39 questions in ASSISTments. The 392 students in the final data set on average have worked in the system for around 7 days (one session per day) with 40 minutes of practice per session. They finished 147 items on average (standard deviation = 60 items) and at maximum 319 items.

In this paper, student models will be built based on the online data to predict student math proficiency at the end of the school year 2004-2005 as measured by the 2005 MCAS test scores. And we will evaluate our prediction against the 2007 MCAS test to answer the research question: Can we model student proficiency as well as the standardized test?

3.2 Modeling

Our effort on predicting student math proficiency starts from **a "lean" model**, an Item Response Theory (IRT)-style ([7], [16]) model within the ASSISTment dataset only. As a start place, we used one-parameter IRT model (the Rasch model[3]), the straightforward model with just student proficiency and item difficulty parameters (and only on original questions). To train the lean model, we used a data set of all data collected in the ASSISTment system from Sept., 2004 to Jan., 2008, including responses to 2,797 items (only on original questions) from 14,274 students. By including more student response data, we hope to acquire a more reliable estimate of the parameters. We fitted the model in BILOG-MG 3.0[4] and added an extra column in the data set that we described in Section 3.1 to record the student proficiency estimates for the 392 students.

[3] In the Rasch model, the probability of a specified response is modeled as a logistic function of the difference between the person and item parameter. In educational tests, item parameters pertain to the difficulty of items while person parameters pertain to the ability or attainment level of people who are assessed.

[4] http://www.scienceplus.nl/scienceplus/main/softwareshop/bilogmg.jsp

The lean model accounts for how students performed on the original questions but totally ignores how students interacted with the system. Much work has been done in the past 10 years or so on developing "online testing metrics" for dynamic testing (or dynamic assessment) [11] to supplement accuracy data (wrong/right scores) for characterizing student proficiency. Researchers have been interested in trying to get more assessment value by comparing traditional assessment (students getting an item marked wrong or even getting partial credit) with a measure that shows how much help they needed. Bryant, Brown and Campione ([4]) compared traditional testing paradigms against a dynamic testing paradigm. Grigorenko and Sternberg ([11]) reviewed relevant literature on the topic and expressed enthusiasm for the idea. In the dynamic testing paradigm, a student would be presented with an item and when the student appeared to not be making progress, would be given a prewritten hint. If the student was still not making progress, another prewritten hint was presented and the process was repeated. In Bryant, Brown and Campione's study they wanted to predict learning gains between pretest and posttest. They found that student's predicted learning gains were not correlated (R = 0.45) with static ability score as well as their "dynamic testing" (R = 0.60) score. It was suggested that this method could be effectively done by computer. Feng, Heffernan & Koedinger ([8]) continued the dynamic testing approach and developed a group of metrics that measures students' accuracy, speed, attempts and help-seeking behavior. We reused these metrics in this study. Namely, the metrics are:

- ORIGINAL_COUNT - the number of original items students have done. This measures students' attendance and on-task-ness. This measure also reflects students' knowledge since better students have a higher potential to finish more items in the same period of time.
- PERCENT_CORRECT - students' percent correct over all questions (both original items and scaffolding questions). In addition to original items, students' performance on scaffolding questions is also a reasonable reflection of their knowledge. For instance, students who failed on original items simply because of their lack of ability of forming problem-solving strategies will probably answer all scaffolding questions correctly.
- QUESTION_COUNT - the number of questions (both original items and scaffolding questions) students have finished. Similar to ORIGINAL_COUNT, this is also a measure of attendance and knowledge but given the fact that scaffolding questions show up only if students failed the original question, it is not obvious how this measure will correlate with students' MCAS scores.
- HINT_REQUEST_COUNT - how many times students have asked for hints.
- AVG_HINT_REQUEST - the average number of hint requests per question.
- HINT_COUNT - the total number of hints students got.
- AVG_HINT_COUNT - the number of hint messages students got averaged over all questions

- BOTTOM-OUT_HINT_COUNT - the total number of bottom-out[5] hint messages students got.
- AVG_BOTTOM_HINT - the average number of bottom-out hint messages students got.
- ATTEMPT_COUNT - the total number of attempts students made.
- AVG_ATTEMPT - the average number of attempts students made for each question.
- AVG_ITEM_TIME - on average, how long does it take for students to finish a problem (including all scaffolding questions if students answered the original questions incorrectly).
- AVG_QUESTION_TIME - on average, how long does it take for a student to answer a question, measured in seconds.
- TOTAL_MINUTES - how many total minutes students have been working on items in the ASSISTment system. Just like ORIGINAL_COUNT, this metric is an indicator of the attendance.

The ten measures from HINT_REQUEST_COUNT to AVG_QUESTION_TIME in the above list are generally all ASSISTment style metrics (or the assistance metrics), which indicate the amount of assistance students need to finish problems and the amount of time they spend to finish items. Therefore, the hypothesis is that all these measures would be negatively correlated with MCAS scores.

Now that we have the online metrics, we ran a backwards regression analysis ($F > .10$) to predict 2005 MCAS test scores. In addition to the metrics themselves, quadratic terms and interaction between the metrics are all introduced into the model initially to address the fact that there may exist non-linear relationships between the test score and the online metrics. We will refer to this model as **the online model.**

We did not include student percent correct on original questions here as it is a static metric that mimics paper practice tests by scoring students as either correct or incorrect on each item. The student proficiency parameter estimated by the lean model will work for the purpose of scoring students plus the advantage that it takes into consideration the fact that item difficulty varies. Therefore, in the next step, we combine the proficiency parameter with the online metrics together and fit another model in SPSS by doing backwards regression ($F > .10$). We name this model **the mixed model**.

Thus, we have constructed three models: the lean model, the online model, and the mixed model and each model addresses different aspects of student performance. In the next section, we will evaluate the models.

[5] Since the ASSISTment system does not allow students to skip problems, to prevent students from being stuck, most questions in the system were built such that the last hint message almost always reveals the correct answer. This message is referred to as "Bottom-out" hint.

3.3 Evaluating results using data from multiple years

Many works (e.g. [1, 2, 8, 9]) have been done to build student models to estimate end-of-year MCAS test scores and predicted scores were compared with actual 2005 MCAS test scores to calculate MAD which was used as the measure to evaluate the student models. In Table 1, we summarize the three regression models and present MAD of each model.

Table 1: Model summary

Model	MAD	R Square	BIC*	#variables entered final model	Correlation with 8th grade MCAS
The lean model	/				.731
The online model	4.76	.786	-437	28	.865
The mixed model	4.61	.794	-470	25	.871

*BIC: Bayesian Information Criteria, using the formula for linear regression models introduced in [13].

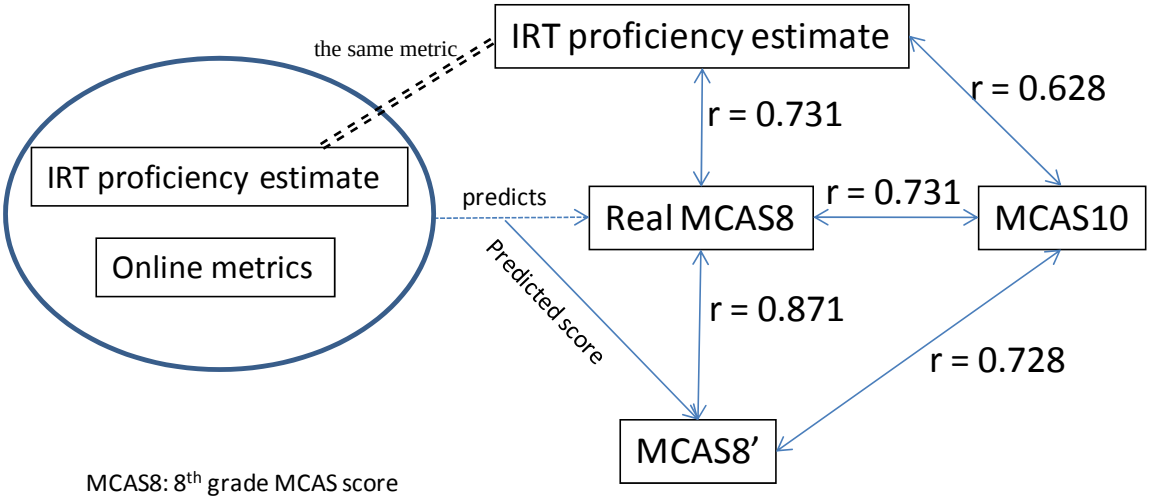
First of all, all of the correlation coefficients in the rightmost column in Table 1 are reliably different from zero, which indicate the data collected within the ASSISTment system can accurately predict student math proficiency at the end of the year. The lean model does not correlate with MCAS score as well as the online model, which indicated that student math proficiency can be better estimated by adding in features reflecting student assistance, effort, attendance, etc. The result is consistent with [8]. Additionally we can improve our prediction of MCAS score further by combining the student proficiency parameter together with the online metrics in the mixed model. The improvement is statistically reliably in terms of BIC[6] while the difference between the correlations is not significant.

A MAD equal to 4.61 is about 8.5% of the total score of the 8th grade MCAS math test, which is not bad. But should we be satisfied or we should push even harder until we can get a MAD of zero? An important question is what a good comparison should be. One reasonable thing to do is to compare to the MCAS, the standardized test itself. Ideally, we wanted to see how good one MCAS test was at predicting another MCAS test. We probably should not hope to do better than that. (But considering there is measurement error in the MCAS test, maybe we can!). We ran a simulation study by randomly splitting one test into two halves and used student scores on one half to predict the second half [9]. They reported MAD of 1.89 that is 11% of the maximum score on half of the test. Since their prediction error in [9] is very close to 11%, they claimed that their approach did as well as MCAS test on predicting math proficiency. In this paper, we propose a different approach to use student cross-year data for determining whether a student model is as good as the standardized test at estimating student proficiency. Assume student math proficiency in the 8th grade and in the 10th grade are highly correlated. Since the

[6] Raftery [[13] discussed a Bayesian model selection procedure, in which the author proposed the heuristic of a BIC difference of 10 is about the same as getting a p-value of p = 0.05

measurement error is relatively independent due to the two years time interval between the tests, therefore, whichever (our student model or the MCAS test) better predicts 10th grade MCAS score is better assessing student math skill at 8th grade.

Let define MCAS8' be the leave-one-out[7] predicted score for 8th grade MCAS that comes from our best student model, the mixed model; MCAS8 be the actual 8th grade MCAS score and MCAS10 be the actual 10th grade MCAS score. Then we asked the question: Can MCAS8' predict MCAS10 better than MCAS8 does? To answer the question, we calculated the correlation between the three metrics: MCAS8', MCAS8 and MCAS10, as presented in Figure 1.



MCAS8: 8th grade MCAS score

MCAS8': our prediction of 8th grade MCAS score using online metrics and student proficiency

MCAS10: 10th grade MCAS score

Figure 1: Correlation between IRT student proficiency estimate, MCAS8', MCAS8 and MCAS10

First of all, we want to point out that all correlations in Figure 1 are statistically reliable ($p < 0.001$). The student proficiency estimated by the lean model correlates with MCAS10 with r equal to .628. It does not do as well as MCAS8 and MCAS8' as we have expected. Even though, we think it is worth finding out and having this lean model, which is based on less data, as a contrast case. It is the most direct test of the question of whether ASSISTment use could essentially replace the 8th grade test. Both MCAS8 and MCAS8' are reliable predictors of MCAS10. MCAS8 correlates with MCAS10 with r equal to 0.731 while the correlation between MCAS8' and MCAS10 is fractionally lower (r = 0.728). A significance test[8] shows they are not statistically reliably different, which suggests that our student model can do as well as MCAS test on predicting the MCAS score two years later. Since both MCAS tests are measuring the student's math proficiency, it can be considered as the evidence that the student model is doing a good job estimating student math proficiency. At the very least, what our system is modeling is relatively stable across a two-year interval.

[7] The adjusted predicted score is calculated by doing "leave-one-out" cross validation in SPSS.

[8] The test is done online at http://www.quantitativeskills.com/sisa/statistics/correl.htm

4 Discussion

There has been a big interest on modeling student knowledge. Corbett & Bhatnagar [6] describes an early and successful effort to increase the predictive validity of student modeling in the ACT Programming Tutor (APT). They used assessments from a series of short tests to adjust the knowledge tracing process in the tutor and more accurately predict individual differences among students in the post test. Beck & Sison [3] used knowledge tracing to construct a student model that can predict student performance at both coarse- (overall proficiency) and fine-grained (for a particular word in the reading tutor) sizes. Anozie & Junker [1] pursued a rather different approach, looking at the changing influence of online ASSISTment metrics on MCAS performance over time. They computed monthly summaries of online metrics similar to those developed in [8], and built several linear prediction models, predicting end-of-year raw MCAS scores for each month. In [8] we developed the metrics as listed in section 3.2 to measure the amount of assistance a student needs to solve a problem, how fast a student can solve a problem, etc. and showed these metrics helped us better assess students. The result in this paper reinforced our previous result as evaluated by a different approach.

In section 3, we describe the method of using student test data from multiple years to compare a student model to a standardized test. Two other approaches have been described in the literature. In [3], Beck & Sison found 3 tests that measures extremely similar constructs to the standardized test that they were interested in. They took the arithmetic mean of those tests as a proxy measure for the true score on the original measure. The pro of this method is that it can be done quickly while the con is that construct validity could be an issue. In [9], we ran a simulation study by "splitting" a standardized test into two parts and the prediction power of the standardized test (actually a half of the standardized test) is determined by how well student performance on one half of the test predicts their performance on the other half. Similarly to the "proxy" measure method in [3], the pro of the "splitting" method is the quickness but it also has some cons. Firstly, if there is measurement error for a particular day (e.g. a student is somewhat ill or just tired), then splitting the test in half will produce a correlated measurement error in both halves, artificially increasing the test's reliability relative to the measure we bring up in this paper (which is not based on data from the same day as the MCAS). Secondly, to do the splitting, it required assess to item level data which is not always available. In this paper, we propose a third method, which is a longitudinal approach. By going across years, we avoid this confound with measurement error, and get a fairer baseline. Though, we do admit that it takes longer time and harder effort to collect data across years (in our case, 3 years).

5 Future work and Conclusions

We will continue working on improving the online assistance metrics. For instance, since the number of hints available is different across problems and the amount of information released in each level of hint differs too, instead of simply summing-up or computing the mean value, we want to construct some weighting function to better measure the amount of assistance students requested to solve a problem. Another piece of work follows up is to predict fine grained knowledge across years. Since our model is clearly capturing

something that is predictive of student future performance, we are considering focusing on determining what predicts specific deficits in an area. The research question we want to answer will be: can an 8th grade student model be used to predict the student will have a problem with a specific 10th grade skill? Teachers will be glad to know the answer so that they can adjust their instruction to better help student knowledge learning.

In this paper, we replicated the study in [8], showing the online ASSISTment metrics are doing a good job at predicting student math proficiency. On top of that, we propose a new method for evaluating the predictive accuracy of a student model relative to the standardized test, using student standardized test scores across years (2005 through 2007). We found some evidence that we can model student math proficiency as well as the standardized test as measured by the new evaluation criterion. Additionally, we want to stress that this is a rather long-term prediction. The collection of the online data started in September, 2004; the 8th grade MCAS score that we are predicting came in at the end of year 2005; while the 10th grade MCAS score that we used to evaluate our prediction were available at the end of year 2007. We consider the new method as a main contribution of this paper as there are few results showing a student model is as good as a standardized test. We have shown that our model hits this level and have presented an alternative way of performing the comparison.

Acknowledgement

This research was made possible by the U.S. Department of Education, Institute of Education Science (IES) grants, "Effective Mathematics Education Research" program grant #R305K03140 and "Making Longitudinal Web-based Assessments Give Cognitively Diagnostic Reports to Teachers, Parents, & Students while Employing Mastery learning" program grant #R305A070440, the Office of Naval Research grant # N00014-03-1-0221, NSF CAREER award to Neil Heffernan, and the Spencer Foundation. All the opinions, findings, and conclusions expressed in this article are those of the authors, and do not reflect the views of any of the funders.

Reference

[1] Anozie N., & Junker B. W. (2006). Predicting end-of-year accountability assessment scores from monthly student records in an online tutoring system. In Beck, J., Aimeur, E., & Barnes, T. (Eds). Educational Data Mining: Papers from the AAAI Workshop. Menlo Park, CA: AAAI Press. pp. 1-6. Technical Report WS-06-05.

[2] Ayers E., & Junker B. W. (2006). Do skills combine additively to predict task difficulty in eighth grade mathematics? In Beck, J., Aimeur, E., & Barnes, T. (Eds). Educational Data Mining: Papers from the AAAI Workshop. Menlo Park, CA: AAAI Press. pp. 14-20. Technical Report WS-06-05.

[3] Beck, J. E., & Sison, J. (2006). Using knowledge tracing in a noisy environment to measure student reading proficiencies. International Journal of Artificial Intelligence in Education, 16, 129-143.

[4] Brown, A. L., Bryant, N.R., & Campione, J. C. (1983). Preschool children's learning and transfer of matrices problems: Potential for improvement. Paper presented at the Society for Research in Child Development meetings, Detroit.
[5] Corbett, A. T., Koedinger, K. R.,& Hadley,W. H. (2001). Cognitive Tutors: From the research classroom to all classrooms. In Goodman, P. S. (Ed.) Technology Enhanced Learning: Opportunities for Change. Mahwah, NJ: Lawrence Erlbaum Associates.
[6] Corbett, A.T. and Bhatnagar, A. (1997). Student modeling in the ACT Programming Tutor: Adjusting a procedural learning model with declarative knowledge. Proceedings of the Sixth International Conference on User Modeling. New York: Springer-Verlag Wein.
[7] Embretson, S. E. & Reise, S. P. (2000). *Item Response Theory for Psychologists*. Lawrence Erlbaum Associates, New Jersey.
[8] Feng, M., Heffernan, N.T., & Koedinger, K.R. (2006a). Addressing the Testing Challenge with a Web-Based E-Assessment System that Tutors as it Assesses. Proceedings of the Fifteenth International World Wide Web Conference. pp. 307-316. New York, NY: ACM Press. 2006.
[9] Feng, M., Heffernan, N.T., & Koedinger, K.R. (2006b). Predicting state test scores better with intelligent tutoring systems: developing metrics to measure assistance required. In Ikeda, Ashley & Chan (Eds.). Proceedings of the 8th International Conference on Intelligent Tutoring Systems. Springer-Verlag: Berlin. pp. 31-40. 2006.
[10] Feng, M. & Heffernan, N. (2007). Towards Live Informing and Automatic Analyzing of Student Learning: Reporting in ASSISTment System. Journal of Interactive Learning Research. 18 (2), pp. 207-230. Chesapeake, VA: AACE.
[11] Grigorenko, E. L. & Sternberg, R. J. (1998). Dynamic Testing. In Psychological Bulletin, 124, pages 75-111.
[12] Olson, L. (2005). Special report: testing takes off. Education Week, November 30, 2005, pp. 10–14.
[13] Raftery, A. E. (1995). Bayesian model selection in social research. In Sociological Methodology, 25, pages 111-163.
[14] Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Koedinger, K. R., Junker, B., Ritter, S., Knight, A., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A., Rasmussen, K.P. (2005). The Assistment Project: Blending Assessment and Assisting. In C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) Proceedings of the 12th International Conference on Artificial Intelligence In Education, 555-562. Amsterdam: ISO Press
[15] Razzaq, Feng, Heffernan, Koedinger, Nuzzo-Jones, Junker, Macasek, Rasmussen, Turner & Walonoski (2007). Blending Assessment and Instructional Assistance. In Nadia Nedjah, Luiza deMacedo Mourelle, Mario Neto Borges and Nival Nunesde Almeida (Eds). Intelligent Educational Machines within the Intelligent Systems Engineering Book Series . pp.23-49. (see http://www.isebis.eng.uerj.br/). Springer Berlin / Heidelberg.
[16] Van der Linden, W. J. & Hambleton, R. K. (eds.) (1997). Handbook of Model Item Response Theory. New York: Springer Verlag.

A Response Time Model for Bottom-Out Hints as Worked Examples

Benjamin Shih, Kenneth R. Koedinger, and Richard Scheines
{shih, koedinger, scheines}@cmu.edu
Carnegie Mellon University

Abstract. Students can use an educational system's help in unexpected ways. For example, they may bypass abstract hints in search of a concrete solution. This behavior has traditionally been labeled as a form of gaming or help abuse. We propose that some examples of this behavior are not abusive and that bottom-out hints can act as worked examples. We create a model for distinguishing good student use of bottom-out hints from bad student use of bottom-out hints by means of logged response times. We show that this model not only predicts learning, but captures behaviors related to self-explanation.

1 Introduction

There are many reasons to measure a student's affective and metacognitive state. These may include adapting instruction to individual needs or designing interventions to change affective states. One technique is to build classification models with tutor interaction data, often using student response times as an independent variable[4, 6, 7, 13]. Several lines of research using this approach have targeted tutor help abuse as a behavior negatively correlated with learning[4, 13], but these classification rules for help abuse can be quite broad and thus proscribe behaviors that may be good for learning. For example, students who drill down to the most detailed hint may not be "gaming" the system, but instead searching for a worked example. The goal of this work is to show that a simple response time-based indicator can discern good bottom-out hint behaviors.

Such an indicator has several primary uses. It may be indicative of general traits, such as good metacognitive self-regulation. It might also be useful as a proxy for student affective states. The indicator, however, might be most useful in improving the design of computer-based educational scaffolding. There is significant evidence to suggest that optimal learning comes from a mix of worked examples and scaffolded problems[11]. A response time indicator for self-explanation behavior can help determine the conditions under which worked examples or scaffolded problems are better.

In this paper, we will discuss the background literature, with a focus on the potential discrepancy between existing tutor models and research in the worked example and self-explanation literature. We will then describe the data used, with an emphasis on the timing information available in the logs. We then focus on a simple student model that leads to an equally simple indicator for good bottom-out hint behaviors. We will demonstrate that the indicator has high correlation with learning gain in the forementioned data, and show

This work was supported in part by a Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (#R305B040063)

that that study's experimental condition provides strong evidence that the indicator does capture some form of reasoning or self-explanation behaviors.

2 Background

Generally, a response time is the time required for a subject to respond to a stimulus. For educational systems, this is often detailed at the transactional level. How long does it take a student to perform a task, such as request help or enter an answer? The limitations of log data means that many details are obscure, such as the student's thought process or the presence of distractions. The student's actions may have been triggered by an event outside the systems' purview or the student may be engaging in other off-task behavior. While a limitation, it is exactly this off-task aspect of response times that make them so valuable for measuring students' affective states[7].

There is prior work on response time-based models for affective and metacognitive states. Beck et. al. modeled student disengagement using a response time-based model[7]. Their work is particulary relevant because, like this study, they use a combination of model building and elimination of irrelevant examples to construct their detector. There are also a number of help-seeking models that utilize response times in their design to better detect "gaming" behavior, particularly behaviors that involve drilling through scaffolding or repeatedly guessing[4, 13]. Still, we have found no examples of models that can detect help abuse aimed at soliciting a worked example. Indeed, one rule shared by many of these models is that bypassing traditional scaffold structure to retrieve the bottom-out hint (which is often the final answer) is considered an undesirable behavior[4]. There are models that attempt to distinguish between positive forms of help abuse and forms of help abuse that negatively impact learning[6]. They do not, however, operate under the same assumptions as this study: that good help abuse potentially involves worked examples and self-explanation.

In the literature on worked examples, particularly in the literature for self-explanation, hint abuse is not always bad for learning. If a student truly needs a worked example, drilling down through the help system may be the easiest available means of getting one. Not only is there plenty of research on worked examples to suggest that this is true[11, 12], there have been attempts to use worked examples in computerized educational systems[5, 10]. Nevertheless, for this study, the focus is on considering worked examples as improving learning through the self-explanation mechanism. Thus, the self-explanation literature provides our hypothesis as to how bottom-out hints might improve learning. In the original self-explanations studies, students were given worked examples and then asked to explain the examples to themselves[8]. Later interventions resulted in improved pre-post gain[9], but extending the self-explanation effect to computerized educational systems has not been straightforward[10]. Using self-explanation interventions in other situations, such as with students explaining their own work, has found some success in computerized systems[3]. This suggests that a more complicated mechanism relates self-explanation behavior to learning in computer tutors. Indeed, a specific connection between bottom-out hints and self-explanation has actually been suggested before[1].

3 Data

Our data is a log of student interactions with the Geometry Cognitive Tutor[2]. In the tutor, students are presented with a geometry problem and several empty text fields. A step in the problem requires filling in a text field. The fields are arranged systematically on each problem page and might, for example, ask for the values of angles in a polygon or for the intermediate values required to calculate the circumference of a circle. In the tutor, a transaction is defined as: interacting with the glossary, requesting a hint, entering an answer, or pressing "done". The done transaction is only required to start a new problem, not to start a new step.

The data itself comes from Aleven et. al.[2] They studied the addition of required explanation steps to the Geometry Cognitive Tutor. In the experimental condition, after entering a correct answer, students were asked to justify their answer by citing a theorem. This could be done either by searching the glossary or by directly inputting the theorem into a text field. The tutor labeled the theorem as correct if it was a perfect, letter-by-letter match for the tutor's stored answer. This type of reasoning or justification transaction is this study's version of self-explanation (albeit with feedback).

The hints for the tutor are arranged in levels, with each later level providing a more specific suggestion on how to proceed on that step. Whereas the early hint levels can be quite abstract, the bottom-out hint does everything shy of entering the answer. The only required work to finish a step after receiving a bottom-out hint is to input the answer itself into the text field. This type of transaction is the study's version of a simple worked example.

There were 39 students in the study that had both pre- and post-test scores. They were split 20-19 between the two conditions. All times were measured in 100th's of a second. The other details of the data will be introduced in the discussion of results.

4 Model

The core assumption underlying this work is that bottom-out hints can serve as worked examples. In our data, a bottom-out hint is defined to be the last hint available for a given step. The number of hints available differs from step to step, but the frequency of bottom-out hint requests suggests that students were comfortable drilling down to the last hint.

Assuming that bottom-out hints might sometimes serve as worked examples, there are a couple ways to detect desirable bottom-out hint behaviors. The first method is to detect when a student's goal is to retrieve a worked example. This method is sensitive to properly modeling student intentions. The second method, which is the focus of this work, is to detect when a student is *learning* from a bottom-out hint. We assume that learning derives from a mechanism similar to self-explanation and that the time spent thinking about a hint is a sufficient measure for that learning. Thus, this approach is applicable regardless of whether the student's intention was to find a worked example or to game the system.

To detect learning from bottom-out hints, our model needs estimates for the time students

Table 1: TER Model

N	Transaction	Cognition	Observed T
1	Request Hint	Think (1)	2.5
2	Try Step	Reflect (1) Think (2) Enter Answer (2)	4.3
3	Try Step	Reflect (2) Think (3) Enter Answer (3)	9.3

Table 2: Hypothetical Student Transactions
(*Observed Data*, Unobserved Data)

Step	Transaction	Observed T	Cognition	Actual T
Step 1	*Request Hint*	*0.347*	...	...
Step 1	*Enter Answer*	*15.152*	Thinking About Hint Looking at Clock Thinking About Date Typing Answer	3 2 9 1.152
Step 2	*Enter Answer*	*4.944*	Thinking About Last Step Thinking About Next Step Typing Answer	2 1 1.944

spend thinking about those hints. Call the hint time HINT_t, where HINT_t is the time spent thinking about a hint requested on transaction t. Estimating HINT_t is nontrivial. Students may spend some of their time engaged in activities unobserved in the log, such as chatting with a neighbor. However, even assuming no external activities or any off-task behavior whatsoever, HINT_t is still not directly observable in the data.

To illustrate, consider the following model for student cognition. On an answer transaction, the student first thinks about an answer, then they enter an answer, and then they reflect on their solution. Call this model TER for Think-Enter-Reflect. An illustration of how the TER model would underly observed data is shown in Table 1. Notice that the reflection time for the second transaction is part of the logged time for the third transaction. Under the TER model, the reflection time for one transaction is indistinguishable from the thinking and entry time associated with the next transaction. Nevertheless, we need an estimate for the Think and Reflect times to understand student learning from bottom-out hints.

The full problem, including external factors, is illustrated in Table 2, which shows a series of student transactions on a pair of problem steps, along with hypothetical, unobserved student cognition. Entries in italics are observed in the log while those in normal face are unobserved, and ellipses represent data not relevant to the example. The time the student spends thinking and reflecting on the bottom-out hint is about 6 seconds, but the only *observed* durations are 0.347, 15.152, and 4.944. In a case like this, the log data's observed response times includes a mixture of Think and Reflect times across multiple steps.

Table 3: TER Model With Estimators

Transaction	Cognition	Notation
Hint	...	...
Enter Answer	Think About Step	K_t
	Off-Task	
	Enter Answer	E_t
Enter Answer	Reflect on Previous	R_t
	Think About New Step	K_{t+1}
	Off-Task	
	Enter Answer	E_{t+1}

Unfortunately, while the reflection time is important for properly estimating HINT$_t$, it is categorized incorrectly. The reflection time for transaction t is actually part of the logged time for transaction $(t+1)$. Teasing those times apart requires estimating the student's time not spent on the hint.

The first piece of our model separates out two types of bottom-out hint cognition: Think and Reflect. Thinking is defined as all hint cognition before entering the answer; reflecting is all hint cognition after entering the answer. Let Think time be denoted K_t and Reflect time be denoted R_t. We define HINT$_t = K_t + R_t$.

The task then reduces to estimating K_t and R_t. As shown earlier, this can be difficult for an arbitrary transaction. However, we focus only on bottom-out hints. Table 3 provides an example of how bottom-out hints differ from other transactions. Note the absence of a Reflect time R_{t-1} in the bottom-out case. Except for time spent on answer entry and time spent off-task, the full time between receiving the hint and entering the answer is K_t. A similar, but slightly more complicated result applies to R_t. For now, assume off-task time is zero - it will be properly addressed later. Let the answer entry time be denoted E_t. Let the total time for a transaction be T_t. Then the equation for HINT$_t$ becomes

$$\begin{aligned} \text{HINT}_t &= K_t + R_t && (1)\\ &= (T_t - E_t) + R_t && (2)\\ &= (T_t - E_t) + (T_{t+1} - (K_{t+1} + E_{t+1})) && (3) \end{aligned}$$

where T_t and T_{t+1} are observed in the log data. The first term consists of replacing K_t with measured and unmeasured times from before the answer is submitted. The second term consists of times from after the answer is submitted. If we have an estimate for E_t, we can now estimate K_t. Similarly, if we have an estimate for K_{t+1} and E_{t+1}, we can estimate R_t.

Constructing reliable estimates for any of the above values is impossible on a per transaction basis. However, if we aggregate across all transactions performed by a given student, then the estimators become more reasonable. There are two other important points regarding the estimators we will use. First, response times, because of their open-ended nature,

are extremely prone to outliers. For example, the longest recorded transaction is over 25 minutes in length. Thus, we will require our estimators be robust. Second, some students have relatively few (≈ 10) bottom-out hint transactions that will fit our eventual criteria. Thus, our estimators must converge quickly.

Now we need some new notation. We will be using the s subscript, where s represents a student. We will also use the $\hat{E}_s$ notation for estimators and the $m(E_t)$ notation for medians. You can think of $m(E_t)$ as approximating the mean, but we will always be using the median because of outliers. E_s, the per student estimator, will represent some measure of the "usual" E_t for a student s. Also let A be the set of all transactions and A_s be the set of all transactions for a given student. Let A_s^1 be the set of all correct answer transactions by a student s where the transaction immediately follows a bottom-out hint. Similarly, let A_s^2 be the set of all transactions that follow a transaction $t \in A_s^1$. For convenience, we will let $T_s^1 = m_{t \in A_s^1}(T_t)$ be the median time for transactions $t \in A_s^1$ and $T_s^2 = m_{t \in A_s^1}(T_{t+1})$ be the median time for transactions $t \in A_s^2$. These two types of transactions are generalizations of the last two transactions shown in Table 3. This gives an equation for our estimator $\hat{\text{HINT}}_s$,

$$\hat{\text{HINT}}_s = (T_s^1 - E_s) + (T_s^2 - (K_s^2 + E_s)) \tag{4}$$

Here, $K_s^2 = m_{t \in A_s^2}(K_t)$ is the thinking time that takes place for transaction $t \in A_s^2$.

Consider $\hat{E}_s$, the median time for student s to enter an answer. It always takes time to type an answer, but the time required is consistently short. If we assume that the variance is small, then $\hat{E}_s \approx \min_{t \in A_s}(E_t)$. That is, because the variance is small, E_t can be treated as a constant. We use the minimum rather than a more common measure, like the mean, because we cannot directly observe E_t. Instead, note that if $K_t \approx 0$, then the total time spent on a post-hint transaction is approximately E_t. Thus, the minimum time student s spends on an answer step is a good approximation of $\min_{t \in A_s}(E_t)$. In practice, the observed $\hat{E}_s$ is about 1 second. With $\hat{E}_s$, we can now estimate K_t for $t \in A_s^1$.

To isolate the reflection time R_s, we need an approximation for K_s^2, the thinking time for transactions $t \in A_s^2$. Unfortunately, K_s^2 is difficult to estimate. Instead, we will estimate a value related to K_s^2. The key observation is, if a student has already thought through an answer on their own, without using any tutor help, they presumably engage in very little reflection after they enter their solution. To put it mathematically, let N_s be the set of transactions for student s where they do *not* use a bottom-out hint. We assume that $R_t \approx 0$, $\forall t \in N_s$. We can now use the following estimator to isolate R_s,

$$\begin{aligned} R_s &= T_s^2 - (K_s^2 + E_s) && (5)\\ &= m(T_t)_{t \in A_s^2} - (K_s^2 + E_s) && (6)\\ &\approx m(T_t)_{t \in A_s^2} - m(T_t)_{t \in N_s} && (7) \end{aligned}$$

where the change from line 6 to line 7 derives from the assumption $R_t \approx 0$, $\forall t \in N_s$. This is the last estimator we require: R_s is approximately $m(T_t - m(T_v)_{(u \in N_s, v = u+1)})_{t \in A_s}$.

Table 4: Indicator Correlations in the Control Condition
(* $p < 0.10$, ** $p < 0.05$)

	Pre	Post	Adjusted Gain
K_s	-0.11	0.37*	0.34*
R_s	0.29	0.42**	0.36*
HINT$_s$	0.04	0.53**	0.48**

That is, we use the median time for the first transaction on a step where the prior step was completed without worked examples. This approach avoids directly estimating K_s^2 and estimates the sum $(K_s^2 + E_s)$ instead.

There is still the problem of off-task time. We have so far assumed that off-task time is approximately zero. We will continue to make that assumption. While students engage in long periods of off-task behavior, we assume that for most transactions, students are on-task. That implies that transactions with off-task behaviors are rare, albeit potentially of long duration. Since we use medians, we eliminate these outliers from consideration entirely, and thus continue to assume that on any given transaction, off-task time is zero.

A subtle point is that the model will not fit well for end-of-problem transactions. At the end of a problem there is a "done" step, where the student has to decide to hit "done". Thus, the model no longer accurately represents the student's cognitive process. These transactions could be valuable to an extended version of the model, but for this study, all end-of-problem transactions will be dropped.

5 Results

We first run the model for students in the control condition. These students were not required to do any formal reasoning steps. The goal is to predict the adjusted pre-post gain, $\max(\frac{\text{(post}-\text{pre)}}{\text{(1}-\text{pre)}}, \frac{\text{(post}-\text{pre)}}{\text{(pre)}})$. We will not use the usual Z-scores because the pre-test suffered from a floor effect and thus the pre-test scores are quite non-normal (Shapiro-Wilks: $p < 0.005$). Two students were removed from the population for having fewer than 5 bottom-out hint requests, bringing the population down to 18. The results are shown in Table 4.

The first result of interest is that none of the indicators have statistically significant correlations with the pre-test. This suggests that they measure some state or trait of the students that is not well captured by the pre-test. The second result of interest is that all three indicators correlate strongly with both the post-test and learning gain. Notably, HINT$_s$, our main indicator, has a correlation of about 0.5 with both the post-test and the learning gain. To the extent that HINT$_s$ does distinguish between "good" versus "bad" bottom-out hint behaviors, this correlation suggests that the two types of behavior should indeed be distinguished.

It's possible that these indicators might only be achieving correlations comparable to time-on-task or average transaction time. As Table 5 shows, this is clearly not the case. All three

Table 5: Time-on-Task Correlations in the Control Condition

	Pre	Post	Adjusted Gain
Time-on-Task	-0.31	-0.10	0.23
Average Transaction Time	-0.03	0.27	0.20

Table 6: Correlations in the Experimental Condition
(* $p < 0.10$, ** $p < 0.05$)

	Pre	Post	Adjusted Gain
K_s	-0.02	0.23	0.25
R_s	0.02	0.31*	0.35*
HINT$_s$	0.00	0.38*	0.41**

hint time indicators out-perform the traditional time-on-task measures.

Nevertheless, these results still do not show whether the indicator HINT$_s$ is actually measuring what it purports to measure: self-explanation on worked examples. For that, we use the experimental condition of the data. In the experimental condition, students are asked to justify their correct solutions by providing the associated theorem. This changes the basic pattern of transactions we are interested in from HINT-GUESS-GUESS to HINT-GUESS-JUSTIFY-GUESS. We can now directly measure R_s using the time spent on the new JUSTIFY steps. R_s is now the median time students spend on a correct justification step after a bottom-out hint, subtracting the minimum time they ever spend on correct justifications. We use the minimum for reasons analogous to those of $\hat{E}_s$ - we only want to subtract time spent entering the reason. In this condition, there were sufficient observations for all 19 students. The resulting correlations are shown in Table 6.

There is almost no correlation between our indicators and the pre-test score, again showing that our indicators are detecting something not effectively measured by the pre-test. Also, the correlations with the post-test and learning gain are high for both R_s and HINT$_s$. While R_s by itself has a statistically significant correlation at $p < 0.10$, K_s and R_s combined demonstrate a statistically significant correlation at $p < 0.05$. This suggests that while some students think about a bottom-out hint before entering the answer and some students think about the hint only after entering the answer, for all students, regardless of style, spending time thinking about bottom-out hints is beneficial to learning. The corollary is that at least some bottom-out hints are proving beneficial to learning.

Thus far, we have shown that the indicator HINT$_s$ is robust enough for strong correlations with learning gain despite being measured in two different ways across two separate conditions. The first set of results demonstrated that HINT$_s$ can be measured without any direct observation of reasoning steps. The second set of results showed that direct observation of HINT$_s$ was similarly effective. Our data, however, allows us access to two other interesting questions. First, does prompting students to explain their reasoning change their bottom-out hint behavior? Second, do changes in this behavior correlate with learning gain?

Table 7: Changes in Behavior in the Experimental Condition
(* $p < 0.10$, ** $p < 0.05$)

	Mean	**Var**	**Pre**	**Post**	**Adjusted Gain**
ΔHINT$_s$	0.56	5.06	0.41*	0.47**	0.38*

To answer both questions, we look at the indicators trained on only the first 20% of each student's transactions. For this, we use only the experimental condition because, when 80% of the data is removed, the control condition has too few remaining students and too few observations. Even in the experimental condition, only 15 remaining students still have more than 5 bottom-out hint requests that meet our criteria. The results are shown in Table 7, with ΔHINT$_s$ representing the difference between HINT$_s$ trained on the first 20% of the data and HINT$_s$ trained on the full data.

To answer the first question, the change in HINT$_s$ is not statistically different from zero. The prompting does not seem to encourage longer response times in the presence of bottom-out hints, so this mechanism does not explain the experimental results of Aleven et. al.'s study[2]. However, some of the students did change their behaviors. As shown in Table 7, students who increased their HINT$_s$ times demonstrated higher learning gain. The evidence is substantial that HINT$_s$ measures an important aspect of student reasoning.

6 Conclusions and Future Work

In this study, we presented evidence that some bottom-out hint use can be good for learning. The correlations between our indicators and pre-post learning gain represent one form of evidence; the correlations between *changes* in our indicators and pre-post learning gain represent another. Both sets of results show that thinking about bottom-out hints predicts learning. However, extending our results to practical use requires additional work.

Our indicators provide estimates for student thinking about bottom-out hints. However, these estimates are aggregated across transactions, providing a student level indicator. While this is useful for categorizing students and offering them individualized help, it does not provide the level of granularity required to choose specific moments for tutor intervention. To achieve that level of granularity, a better distributional understanding of student response times would be helpful, as would an indicator capable of distinguishing between students seeking worked examples versus engaging in gaming. Exploring how the distribution of response times differs between high learning bottom-out hint students and low learning bottom-out hint students would go a long way to solving both problems.

That issue aside, our indicators for student self-explanation time have proven remarkably effective. They not only predict learning gain, they do so better than traditional time-on-task measures, they are uncorrelated with pre-test scores, and changes in our indicators over time also predict learning gain. These indicators achieve this without restrictive assumptions about domain or system design, allowing them to be adapted to other educational systems in other domains. Whether the results transfer outside of geometry or to other

systems remains to be seen, but they have so far been robust.

The inclusion of two conditions, one without justification steps and one with justification steps, allowed us to show that the indicators do measure something related to reasoning or self-explanation. We estimated the indicators for the two conditions in different ways, yet both times, the results were significant. This provides a substantial degree of validity. However, one direction for future work is to show that the indicators correlate with other measures of self-explanation or worked example cognition. One useful study would be to compare these indicators with human estimates of student self-explanation.

References

[1] Aleven, V., Koedinger, K. R. Limitations of student control: Do students know when they need help? *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, 2000, p. 292–304.

[2] Aleven, V., Koedinger, K. R. An effective meta-cognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science*, 26(2), 2002, p. 147–179.

[3] Aleven, V., Koedinger, K. R., Cross, K. Tutoring answer explanation fosters learning with understanding. *Proceedings of the 9th International Conference on Artificial Intelligence in Education*, 1999.

[4] Aleven, V., McLaren, B. M., Roll, I., Koedinger, K. R. Toward tutoring help seeking - applying cognitive modeling to meta-cognitive skills. *Proceedings of the 7th Conference on Intelligent Tutoring Systems*, 2004, p. 227–39.

[5] Atkinson, R. K., Renkl, A., Merrill, M. M. Transitioning from studying examples to solving problems: Effects of self-explanation prompts and fading worked-out steps. *Journal of Educational Psychology*, 95(4), 2003, p. 774–783.

[6] Baker, R. S., Corbett, A. T., Koedinger, K. R. Detecting student misuse of intelligent tutoring systems. *Proceedings of the 7th International Conference on Intelligent Tutoring Systems*, 2004, p. 531–540.

[7] Beck, J. Engagement tracing: using response times to model student disengagement. *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, 2005, p. 88–95.

[8] Chi, M. T. H., Bassok, M., Lewis, M., Reimann, P., Glaser, R. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13, 1989, p. 145–182.

[9] Chi, M. T. H., de Leeuw, N., Chiu, M. H., LaVancher, C. Eliciting self-explanations improves understanding. *Cognitive Science*, 18, 1994, p. 439–477.

[10] Conati, C., VanLehn, K. Teaching meta-cognitive skills: Implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. *Proceedings of the 9th International Conference on Artificial Intelligence in Education*, 1999, p. 297–304.

[11] Paas, F. G. W. C., Van Meeriënboer, J. J. G. Variability of worked examples and transfer of geometrical problem-solving skills: A cognitive-load approach. *Journal of Educational Psychology*, 86(1), 1994, p. 122–133.

[12] Sweller, J., Cooper, G. A. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2, 1985, p. 251–296.

[13] Walonoski, J. A., Heffernan, N. T. Detection and analysis of off-task gaming behavior in intelligent tutoring systems. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 2006, p. 722–724.

Mining Student Behavior Models in Learning-by-Teaching Environments

Hogyeong Jeong and Gautam Biswas
{hogyeong.jeong, gautam.biswas}@vanderbilt.edu
Department of Electrical Engineering and Computer Science, Vanderbilt University

Abstract. This paper discusses our approach to building models and analyzing student behaviors in different versions of our learning by teaching environment where students learn by teaching a computer agent named Betty using a visual concept map representation. We have run studies in fifth grade classrooms to compare the different versions of the system. Students' interactions on the system, captured in log files represent their performance in generating the causal concept map structures and their activities in using the different tools provided by the system. We discuss methods for analyzing student behaviors and linking them to student performance. At the core of this approach is a hidden Markov model methodology that builds students' behavior models from data collected in the log files. We discuss our modeling algorithm and the interpretation of the models.

1 Introduction

We have been developing computer environments for middle school students that use the learning by teaching paradigm to help middle school students learn metacognitive strategies while learning in science domains [1]. To teach, one must gain a good understanding of the domain knowledge and then structure that knowledge in a form that they can present to others [8]. Preparing to teach is a self-directed and open-ended activity where one explores, integrates, and structures knowledge first for oneself, and then for others. In addition to preparatory activities, teachers answer questions, provide explanations during teaching, and receive feedback from their students.

In our teachable agent (TA) environment, students teach a computer agent using well-structured visual representations. Using their agent's performance (which is a function of how well it was taught) as a motivation, students work to remediate the agent's knowledge, and, in this process, they learn better on their own [1]. An important property of our TA environments is that students can monitor how their agents answer questions and solve problems, and they may correct them when they notice discrepancies between their own knowledge and the agent's. Moreover, students' errors come more from a conflict between students' background knowledge and the procedures involved than the difficulty of using such a system [2]. Therefore, these environments (provided that they are well-designed) could help students become more knowledgeable of and responsible for their own cognition and reasoning. As a result, students are likely to develop problem solving and monitoring skills that go beyond the learning of domain specific content; rather these environments provide the much larger framework that guide students on *how to learn* and how to *prepare for future learning* [10].

We have run a number of experimental studies to understand the cognitive and metacognitive processes that help students learn with understanding in the TA environments. The

studies have contrasted situations where students either learn by teaching or are taught by a mentor agent. In one condition, students receive metacognitive prompts as they learn by teaching. This paper investigates the use of data mining methods to systematically build and analyze models of student behaviors as they work in the TA environments. Traditional student modeling methods have focused on capturing students' knowledge of the domain. In contrast, our emphasis on preparation for future learning has directed us to focus on strategies and behaviors that students employ during the learning process. This paper approaches student modeling by analyzing student behaviors from their activity sequences, and linking these behaviors to learning strategies.

1.1 Learning by teaching: The Betty's Brain system

The Betty's Brain system is illustrated in Fig. 1. The teaching process is implemented as three primary activities: (i) *teach* Betty concepts and links using a concept map representation [6]; (ii) *query* Betty to find out how she answers questions using what she has been taught; and (iii) *quiz* Betty on a set of predefined questions generated by the mentor agent to see how she performs.

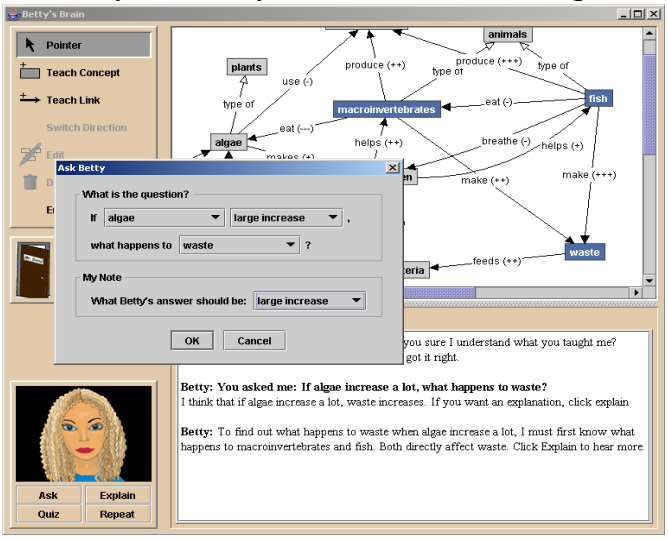


Figure 1: Betty's Brain System with Query Window

Betty uses qualitative reasoning methods to reason through chains of links to answer questions, and, if asked, explains her reasoning using text and animation schemes. She also provides feedback that reflects the students' teaching behaviors. The goal is to get the students to adopt metacognitive strategies in their learning tasks [11]. Students reflect on Betty's answers and her explanations, and revise their own knowledge as they make changes to the concept maps to teach Betty better.

1.2 Experimental Design

Our participants were 56 students in two 5th grade science classrooms taught by the same teacher. Data for three students was dropped from the main study, and three more were dropped from the transfer study due to excessive absences. Students were assigned to one of three conditions using stratified random assignment based on standardized test scores. The students first created concept maps on river ecosystem concepts and causal relations in the main phase (seven 45-minute sessions) of the study. Depending on their assigned group, students utilized one of three versions of the system: a) a learning by teaching (LBT) version in which students taught Betty, b) a self-regulated learning by teaching (SRL) version in which students taught Betty and received metacognitive prompts from Betty, and c) an intelligent coaching system (ICS) version in which students created a map for themselves with guidance from the mentor agent. The ICS system served as our control condition [3]. After an eight-week delay, students participated in the transfer

phase (five 45-minute sessions) in which they learned about a new domain, the land-based nitrogen cycle. During this phase, all students used a stripped-down version of the LBT system in which all of the feedback provided by Betty or the mentor was removed.

1.3 Results

We scored the students' final main and transfer concept maps to identify correct inclusions of concepts and links based on the resources that were provided to the students. Table 1 shows the average concept map scores by condition for the two phases of the study. Students who taught developed more complete and interconnected concept maps than students who created maps for themselves. The differences in map scores are statistically significant (SRL > LBT, ICS; LBT > ICS; $p < 0.05$) in the main phase, and the difference between SRL and ICS persisted during the transfer phase.

Table 1. Concept map score: main and transfer phases

Condition	Mean (SD) Map Scores	
	Main Phase	**Transfer Phase**
ICS (n = 17)	22.83 (5.3)	22.65 (13.7)
LBT (n = 15)	25.65 (6.5)[c]	31.81 (12.0)
SRL (n = 18)	31.58 (6.6)[a,b]	32.56 (9.9)[a]

[a] SRL > ICS, $p < .05$; [b] SRL > LBT, $p < .05$; [c] LBT > ICS, $p < .05$.

We also performed preliminary analyses of students' behavior sequences to shed light on their learning processes [1]. We found that students who generated better concept maps used balanced learning strategies: they read resources, built and edited their concept maps, asked queries to probe the correctness of their maps, and used quiz questions in equal measure to learn about the domain. On the other hand, students who generated low scoring concept maps adopted a myopic learning strategy that overly focused on getting their quiz answers correct. They relied mostly on the quiz questions, and did not seem to read resources or probe their maps by asking queries very often.

1.4 Model Selection

To investigate this relation between learning performance and the use of strategies by group, it became important to go beyond frequency counts or proportions of individual activities, and examine how these activities came together as larger behavior patterns and strategies. To this end, we found the hidden Markov models (HMMs) to be the most appropriate, as they allow us to identify some of the students' general behavior patterns from sequences of their interactions with the system..

2 A Data Mining Approach to Analyzing Student Behaviors

Our approach to analyzing student behaviors in the main and transfer phases of the study involves four steps that appear in most data mining applications [12]: (i) devise a logging system that records student interactions with the system; (ii) perform data cleaning by parsing the generated log files and splicing the information into desired activity sequence data that will form the input to the HMM generating algorithm; (iii) construct the HMM

models; and (iv) interpret generated models as student learning behaviors and compare models across conditions. We describe each of these steps in greater detail below.

2.1 Generating the Raw Data: The Logging System

The raw data corresponds to actions performed by students on the system and the responses provided by Betty and the mentor agent. The agents and the student interact by communicating through the system environment agent. For example, if the student asks Betty a question, the request is broadcast to the environment agent, who routes it to other agents. The left side of Fig. 2 illustrates the message passing that takes place in the system. The signal first broadcast to the environment agent is routed to the teachable agent, who makes a request to the environment agent to use the current concept map to answer the question. This raw data is then processed in real-time and stored in log files as shown in the right side of Fig. 2. At the end of a session, the logs are sent to a server and consolidated into a session by session sequence for each student into a database.

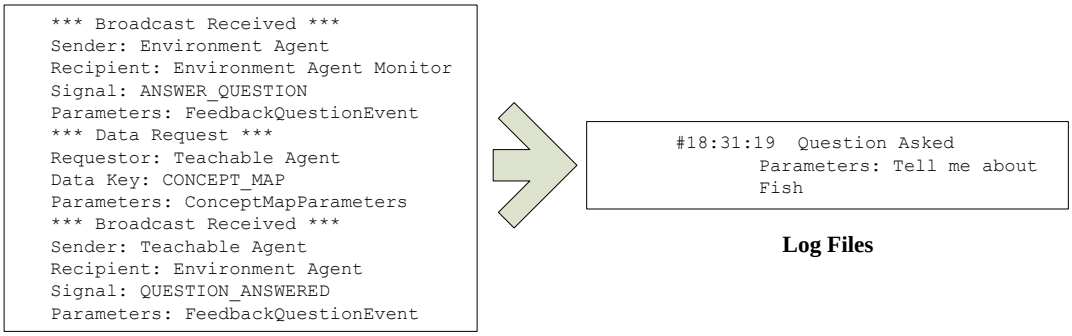


Figure 2: Events and messages being turned to log files

2.2 Parsing the Log Files

In this study, we derive students' behavior patterns by analyzing the sequence of their interactions with the system over multiple sessions. To simplify the interpretation task we lumped analogous student actions into one aggregate activity. For example, all of the map

Table 2: Student Activities and Related Actions

Activity	Student Actions
Edit Map (EM)	adding, modifying, or deleting concepts and links
Ask Query (AQ)	asking Betty queries
Request Quiz (RQ)	asking Betty to take the quiz
Resource Access (RA)	accessing the resources
Request Explanation (RE)	asking Betty for an explanation to her query answer
Continue Explanation (CE)	asking Betty to provide a more detailed explanation
Quiz Denied (QD)	asking Betty to take the quiz without adequate preparation
Quiz Taken (QT)	asking Betty to take the quiz with adequate preparation

creation and modification activities, i.e., adding concepts and links, deleting concepts and links, and editing nodes and links, were combined into one aggregate activity called Edit Map (EM). All student activities were expressed as the eight activities summarized in Table 2. Examples of the resultant sequences are shown in Fig. 3 below.

```
S5213,1,RA,EM,EM,EM,EM,EM,EM,AQ,AQ,AQ,EM,EM,AQ,EM,EM
S5254,1,RA,RA,RA,RA,EM,EM,EM,AQ,AQ,EM,EM,EM,EM,EM,EM,EM
S6236,1,RA,EM,EM,EM,AQ,EM,EM,EM,EM,EM,EM,EM,EM,EM,EM,EM,RQ,QT,EM,EM,AQ,AQ
```

Figure 3: Parsed Data for 3 students in session 1

2.3 Constructing the HMMs

The first step in interpreting this behavior data was to generate hidden Markov models. A hidden Markov model consists of hidden states that are not directly visible, and are governed by three sets of parameters: initial probability vector π, transition probability matrix, *A*, and output probability matrix, *B* [7]. By representing concise models of student activity patterns, a hidden Markov model has the potential of providing us with a global aggregated view of how students approach the learning task [3].

The algorithm that constructs HMMs given a set of observation sequences derives an optimal set of these parameters (π, *A, B*) that maximizes the likelihood of the input sequences. Further, simpler models are easier to interpret (Occam's razor principle), so we apply an algorithm developed by Li and Biswas [5] that uses the Bayesian information criterion (BIC) to trade off simplicity of the model against information provided by the model. BIC, defined as $\log(L) - d/2\log(N)$, uses the log likelihood of the model ($\log(L)$), model size (d), and the number of observations (N) to find the model that strikes a balance between high likelihood and low complexity [5].

Finding the optimal HMM parameters from data is an optimization problem. Two common iterative convergence optimization schemes are the Baum-Welch [7] and the segmental K-Means [4] algorithms. In this paper, we use the segmental K-Means algorithm in conjunction with BIC for iterative segmentation and optimization steps to achieve the optimal model parameters, which include (π, *A, B*) and the number of states in the model, *d*. The segmentation step uses the Viterbi algorithm for sequential decoding, while the optimization step finds a new set of model parameters as dictated by the K-Means method [4]. A chief advantage of the K-Means algorithm is its faster execution time gained by setting a restricted optimization objective. In the future, the faster speed may allow online computation of the behavior models to provide guided feedback as the student works on the system.

A concern, however, for the segmental K-Means algorithm is the likelihood of its convergence to local maxima. In this work, we ran the algorithm one hundred times with random initializations (by sampling the initial parameter values from uniform distributions), all of which converged to the same configuration. We also compared the BIC values generated by the Baum-Welch and the segmental K-Means algorithms, and found that the K-Means algorithm produced consistently better results (see Fig. 4). While these empirical results do not conclusively dispel the fact that our solutions may converge to local maxima, they nevertheless show the algorithm to be quite robust.

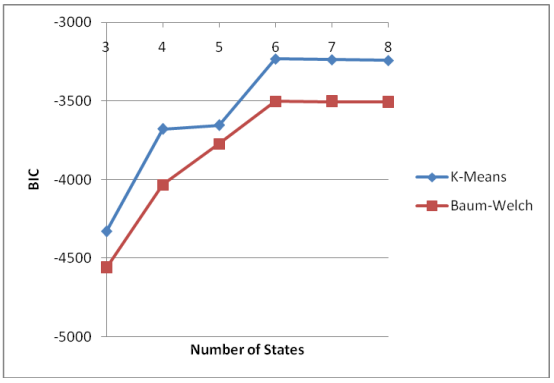


Figure 4: K-Means and Baum-Welch-generated BIC values for the ICS Model

The parsed activity sequences were used to derive two sets of three hidden Markov models for the three conditions using the above algorithm.

3 Analysis of the HMMs

The behavior sequence models for the ICS, LBT, and SRL groups in the main and transfer study created using our HMM algorithm are shown in Fig. 5. Each model is made up of a set of states, the activity patterns (the output probability) associated with each state, and the transition probabilities between states. For example, the model predicts that students in the ICS condition RQ(75%)CE(25%) state requested the quiz 75% of the time, and asked for a continued explanation 25% of the time. The transition probability associated with a link between two states indicates the likelihood of the student transitioning from the current state to the indicated state. For example, the HMM model states student in the ICS condition in state RQ(75%)CE(25%) of the main phase would demonstrate an 6% likelihood of transitioning to state EM, a 19% likelihood of remaining in the same state, and 71% likelihood of transitioning to state QT. Likelihoods less than 2% were not represented in the figure, explaining why these numbers do not sum to 100%. HMMs are so named because their states are hidden. That is, they are not directly observed in the input sequences, but provide an aggregated description of the students' interactions with the system. Sequences of states may be interpreted as the students' learning behavior patterns. We investigate further by interpreting these models in terms of cognitive learning behaviors of the students.

In looking at these HMMs, we find several interpretable patterns that present themselves through high intrastate transition probabilities and low interstate transition probabilities. For example, we see that the transition probabilities from states with Request Explanation (RE) to Continue Explanation (CE) are strong (≥49%). Also, we see that these states are quite isolated, as the transition probabilities into these two states from other states are typically small (only 7% in the transfer SRL model). We combine these observations with knowledge of patterns relevant to interactive metacognition to find three patterns: *basic map building*, *map probing*, *and map tracing* [3].

Basic map building is a pattern characterized by editing the map, submitting the map for a quiz, and occasionally referring to the reading resources. The pattern reflects a basic and important metacognitive strategy. Students work on their maps, check the map by taking a quiz to see if there are flaws, and occasionally refer to the readings.

Map probing is defined by students asking questions of their map to check for specific relations between two concepts (e.g., if fish increase, what happens to algae?). This pattern exhibits a more proactive, conceptually driven strategy, because students are targeting specific relations rather than relying on the quiz to identify errors. Students also need to formulate their own questions to do so.

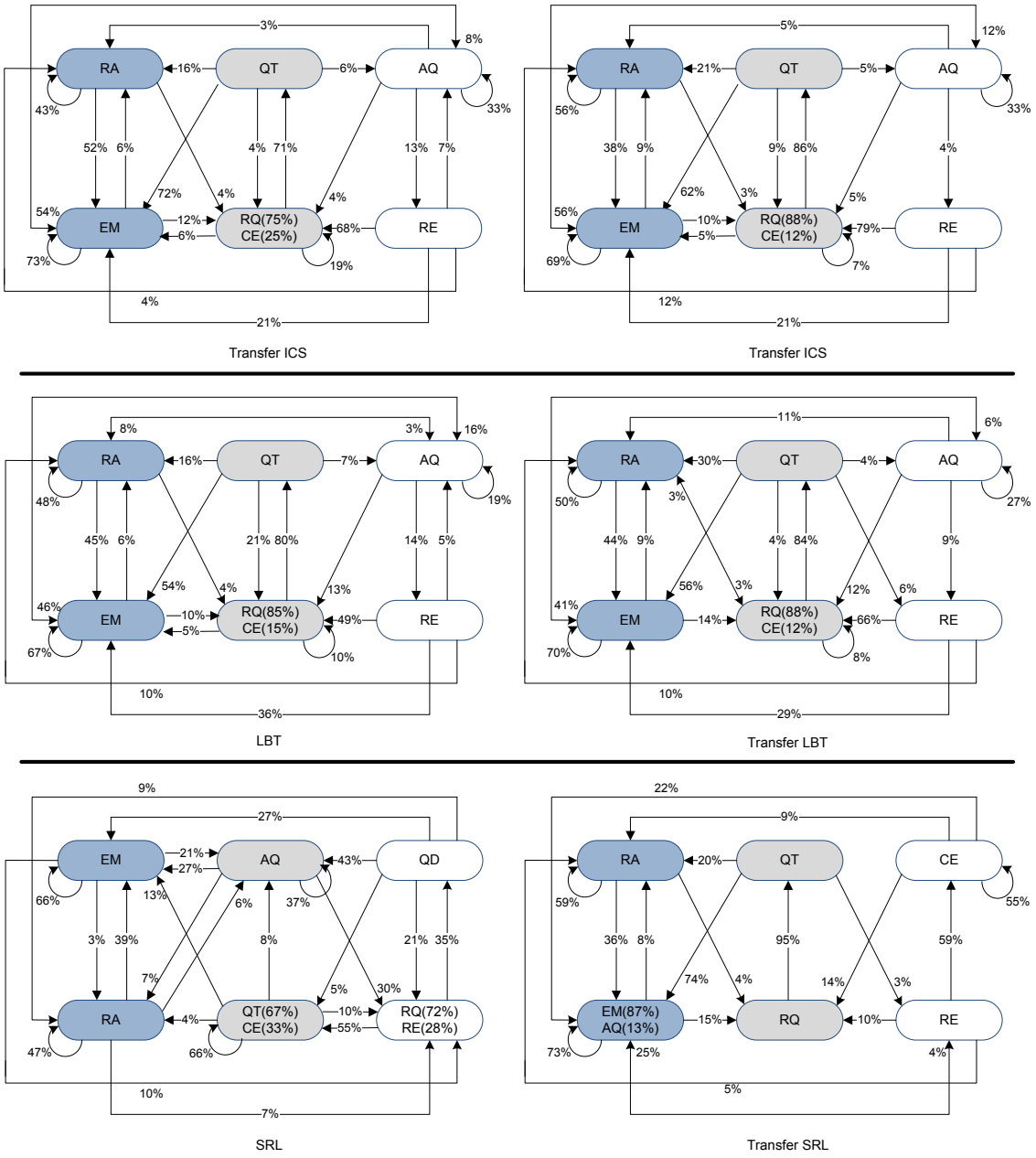


Figure 5: HMM Models for the three conditions in the main and transfer phase

Map tracing pattern reflects students asking Betty or the mentor (depending on the system) to explain the reasoning step by step. When Betty or the mentor initially answers a question, they state that a change in one entity causes a change in another entity and highlight the paths they followed to reach their answer. To follow the details of the inference chain, students had to ask Betty or the mentor agent to explain their reasoning. The agents did so by hierarchically decomposing the chain of inference; for each explanation request, they showed how a particular path within the larger chain contributed to the final answer. Receiving more details about the reasoning process is particularly useful when maps become complex, and there are multiple paths between two concepts.

To build reduced versions of HMMs that incorporate these patterns, we first built aggregate states that represented the patterns of its individual behaviors. For instance, Edit Map, Request Quiz, Quiz Taken, Quiz Denied, and Resource Access were combined into the basic map building state; Ask Query was treated as map probing state; and Request and Continue Explanation were combined into a map tracing state. The transitions were then constructed in accord to a proportional weighing of the individual behavior's stationary probabilities.

3.1 Initial Analysis

Our preliminary analysis consisted of examining the prevalence of each behavior in the resultant stationary probabilities. The derived stationary probability values are listed in Table 3. In a sense, this analysis is equivalent to the frequency count analysis that we have performed in other studies [3], and indicates an estimate of the relative time spent in each state. Similar results in both studies help to validate our methods and results.

Table 3: Grouped Stationary Probabilities

State	ICS	LBT	SRL	Transfer ICS	Transfer LBT	Transfer SRL
Map Building	0.72	0.66	0.42	0.73	0.73	0.68
Map Probing	0.24	0.30	0.47	0.25	0.25	0.27
Map Tracing	0.04	0.04	0.11	0.02	0.02	0.05

In the main phase, we see that the differences in stationary probabilities among the groups are quite pronounced. For example, we see that there exists a significant drop-off in the probabilities of the students' edit map behaviors between successive conditions. Meanwhile, we see proportional increases in activities belonging to higher-level patterns, such as requesting explanation, and continuing explanation. The students' use of continue explanation is especially pronounced.

In the transfer phase when the students operate in a common environment, we see that the differences become smaller. In all three groups, we see a great spike in the number of resource accesses relative to the main phase. At the same time, we observe a decrease in the occurrence of some of the higher-level patterns. This may be due to the students learning about a new domain with a simpler expert map that contains fewer interdependences between concepts, and being given few sessions than in the main phase (five versus seven). This also could be due to the students internalizing the reasoning mechanism, therefore, a reduced need to perform such activities in the transfer phase [3].

3.2 Analysis of the HMM Patterns

Our next level of analysis consisted of examining the interactions among the metacognitive states and their transitions in our models (Fig. 6). These interactions inform us about students' typical learning behavior patterns by condition that we have identified.

In the main phase, we find that the students in the ICS group tend to stay mainly in the basic map building state, while the SRL students tend to stay more in the higher-level states once they get there. For example, the self-loop vector for the map tracing state is much larger in the SRL group (23%) than in the ICS or the LBT groups (9% and 5% per-

cent, respectively). Also, while the differences between ICS and LBT seem to be small, the ICS students seem spend most of their effort in basic map building, while the LBT students do spend a little more time in map probing, a higher level metacognitive activity.

In the transfer phase, the difference between the ICS and the LBT group becomes harder to discern. However, like the main phase, the LBT students seem more likely to enter map tracing state than the ICS students (5% as opposed to 3%), but are more likely to leave once they get there. However, unlike in the main phase, the LBT students now seem to be more confined to the basic map building state than the ICS students (83% as opposed to 81%). However, the SRL students still perform more map probing than the other groups.

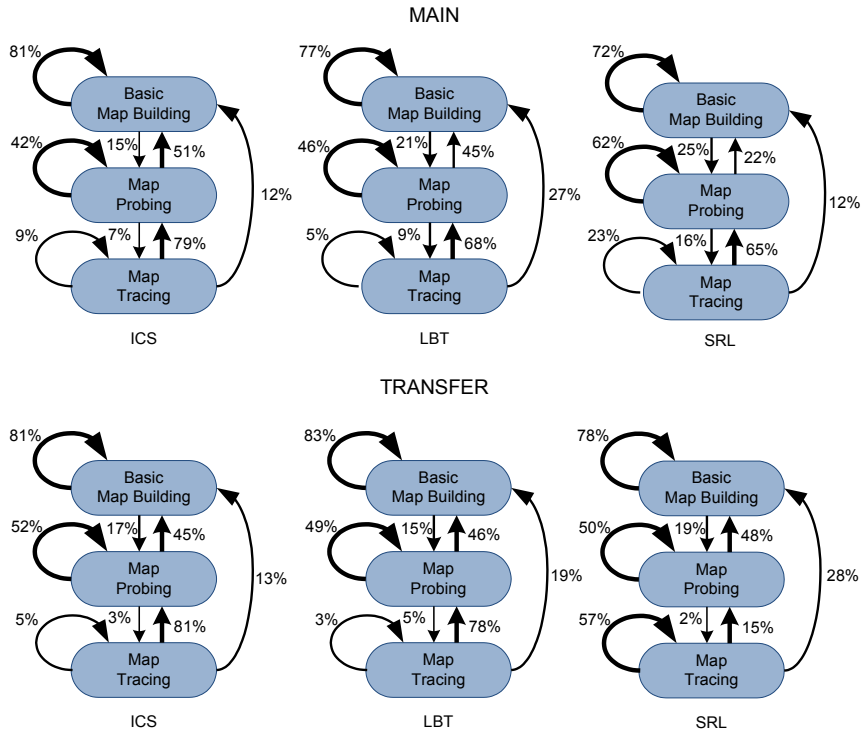


Figure 6: HMM Patterns showing ICS, LBT and SRL behaviors in the main and transfer study

4 Conclusions and future work

Our data mining approach to building HMM models of student behaviors from log files have been very revealing. They have helped us establish that learning by teaching provides better opportunities for learning even among 5th grade students. Further, metacognitive prompts while learning enable students to develop higher level learning strategies that they retain even when the feedback prompts are removed. In the future, we will further refine our data mining techniques and algorithms to set up a framework for designing adaptive learning environments, where the learning support and feedback provided to students will be guided by the information derived from the student behavior models. We will also work further towards developing a more quantitative way of analyzing and

comparing the models (these may involve using distance metrics and more comprehensive cognitive learning patterns).

Acknowledgements

This work has been supported by a Dept. of Education IES grant #R305H060089 and NSF REESE Award #0633856. The authors wish to acknowledge the help provided by John Wagster and Rod Roscoe in the data collection, data analysis, and data interpretation tasks.

References

[1] Biswas, G., Leelawong, K., Schwartz, D., Vye, N., The Teachable Agents Group at Vanderbilt. Learning by Teaching: A New Agent Paradigm for Educational Software. *Applied Artificial Intelligence*, 2005, 19, p. 363-392.

[2] Easterday, M., Aleven, V., and Scheines R. The logic of Babel: Causal reasoning from conflicting sources. Artificial Intelligence in Education, Marina del Rey, California, 2007.

[3] Jeong, H., Gupta, A., Roscoe, R., Wagster, J., Biswas, G., Schwartz, D. Using Hidden Markov Models to Characterize Student Behavior Patterns in Computer-based Learning-by-Teaching Environments. To appear in: *Intelligent Tutoring Systems*, Montréal, Canada, 2008.

[4] Juang B., Rabiner L. The segmental k-means algorithm for estimating the parameters of hidden Markov Models. *IEEE Trans. Acoustics, Speech and Signal Processing*, 1990, 38(9) p. 1639-1641.

[5] Li, C. and Biswas, G. A Bayesian Approach for Learning Hidden Markov Models from Data. *Special issue on Markov Chain and Hidden Markov Models, Scientific Programming*, 2002, 10, p. 201-219.

[6] Novak, J.D. Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations. *Lawrence Erlbaum Associations*, 1998. Mahwah, NJ.

[7] Rabiner L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE*, 1989, 77(2), p. 257-286.

[8] Roscoe, R. D., Chi, M. Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of Educational Research*, 2007, 77(4), p. 534-574.

[9] Roscoe, D., Wagster, J., and Biswas, G., Using Teachable Agent Feedback to Support Effective Learning by Teaching, in review, Cognitive Science Conference, Washington, DC, July 2008.

[10] Schwartz, D.L. and Martin, T., Inventing to Prepare for Future Learning. *Cognition and Instruction*, 22(2), p. 129-184, 2004.

[11] Wagster, J., Tan, J., Wu, Y., Biswas, G., Schwartz, D. Do Learning by Teaching Environments with Metacognitive Support Help Students Develop Better Learning Behaviors? *The twenty-ninth Annual Meeting of the Cognitive Science Society, Nashville, Tennessee*, 2007, p. 695-700.

[12] Whitten, I.H. and Berry, M., Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed, Morgan Kaufmann., Los Altos, CA, 2005.

Argument Graph Classification via Genetic Programming and C4.5

Collin Lynch[1], Kevin D. Ashley[1,2], Niels Pinkwart[3], and Vincent Aleven[4]
collinl@cs.pitt.edu, ashley@pitt.edu, niels.pinkwart@tu-clausthal.de, aleven@cs.cmu.edu
[1] Intelligent Systems Program, University of Pittsburgh.
[2] School of Law, University of Pittsburgh.
[3] Department of Informatics, Clausthal University of Technology.
[4] HCII, School of Computer Science, Carnegie Mellon University.

Abstract. In well-defined domains there exist well-accepted criteria for detecting good and bad student solutions. Many ITS implement these criteria characterize solutions and to give immediate feedback. While this has been shown to promote learning, it is not always possible in ill-defined domains that typically lack well-accepted criteria. In this paper we report on the induction of classification rules for student solutions in an ill-defined domain. [1] We compare the viability of classifications using statistical measures with classification trees induced via C4.5 and Genetic Programming.

1 Introduction

Much of the success of Intelligent Tutoring Systems stems from their ability to give on-line assessment and feedback. This assessment is usually based upon the implementation of *a-priori* domain or tutoring knowledge. Such implementation works in a well-defined domain where there is widespread agreement about the structure of domain knowledge, relevant solution characteristics and acceptable solution processes.

In ill-defined domains [9], while it is possible to identify individual characteristics that are endorsed or proscribed by some domain experts, it is difficult to find widespread agreement about atomic solution actions let alone whole solutions or problem solving behaviors. This is a fundamental challenge for automating effective instruction in ill-defined domains.

In recent years the Educational Data-Mining (EDM) community has sought to augment the *a-priori* knowledge in existing tutoring systems with more automatically derived knowledge. Work by Baker et al. on gaming detection [1], Cen et al. on over-training [3], and Feng et al. on student modeling [5] has shown the utility of EDM to induce new pedagogical information. However, this work has been based upon statistical learning methods which, while they yield successful predictors, do not permit detailed examination of their internal logic to induce domain knowledge. In a word, they are not easily "inspectable".

Work by Harrer et al. [6] and Miksatko and McLaren [11] has shown the utility of user-guided extraction of inspectable solution patterns. This work, has focused on a manual approach or a model-driven "guided-discovery" approach where user specified patterns are explored in the dataset to highlight interactions.

[1]This research is supported by NSF Award IIS-0412830.

In this paper we describe our work on the automatic extraction of informative behavior patterns from student solutions in the ill-defined domain of legal argumentation. This work is based upon our existing tutoring system for law and legal argumentation called LARGO. As described in the next section, LARGO guides students through the process of analyzing and annotating Supreme Court oral arguments using *a-priori* structural hints. While these hints have been shown to be beneficial, they are restricted to relatively small-scale solution characteristics and, due to the limits of our domain knowledge, do not enable us to efficiently characterize the student solutions as a whole.

Our goal in this work is to induce inspectable classifiers capable of predicting a student's post-test performance based upon their solution characteristics. We will then analyze these classifiers comparing them to our domain model, checking our expectations of good student behavior and identifying similar groups of poor performers. For the present we chose to induce binary decision trees via C4.5 and Genetic Programming. We felt that these structures were expressive enough to cover a range of student behaviors while inspectable enough to be analyzed by domain experts and rendered into instructional explanations. We begin by providing the necessary background on LARGO, C4.5. and Genetic Programming. We follow this with a discussion of the rules that were induced and their relative successes and failures. We then conclude with a discussion of the suitability of these techniques for future work.

2 Background

LARGO, an Intelligent Tutoring System for legal argumentation [15], supports students in the reading, annotation, and analysis of oral argument transcripts taken from the U.S. Supreme Court. These transcripts cover cases argued before the court and contain complex real-world examples of legal argument as performed by practicing experts.

These arguments are characterized by the use of proposed tests and hypothetical cases. Advocates before the court propose a test or legal rule about how the case should be decided. Not coincidentally the outcome of this rule favors their client. Judges in turn probe the test by posing hypothetical cases, variations on the specific facts of the case that stress one part of the test or another to probe for weaknesses. The advocates respond by reformulating the test to take into account the hypotheticals or distinguishing the hypothetical from the case at hand, citing relevant facts and legal principles with which to justify their argument.

LARGO is designed to facilitate students' understanding of this form of argument by supporting their analysis of the examples. Students using LARGO are presented with a series of case problems comprising oral arguments to annotate using a graphical markup language. This markup is composed of test, hypothetical and fact nodes, as well as arcs connecting them. Students may add descriptive labels to any of the nodes and arcs, and may link the Test and Hypo nodes to selected portions of the argument transcript. The same formalism and the characteristics described below can also be used for the production of novel arguments in a manner similar to [4]. A sample diagram is shown in Figure 1.

LARGO analyzes the student diagrams for structural, contextual and content-related "characteristics" which we use as the basic features of our current analysis. Each characteristic is defined by a particular graphical pattern that, if it matches some portion of a student's diagram, identifies a possible structural weakness or opportunity for reflection. The characteristics were developed

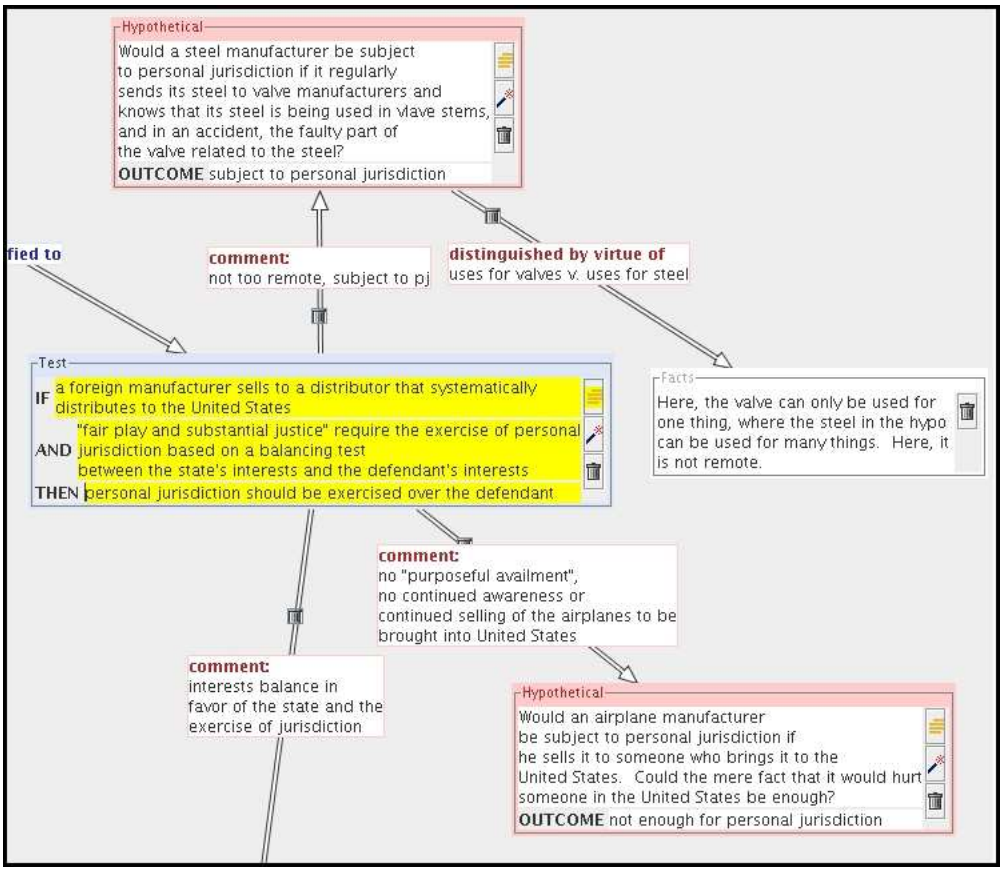


Figure 1. Sample LARGO Graph

with the help of an experienced legal instructor. For example *UNLINKED_TEST*, a context characteristic is active when the student has formed a test box in the graph but has not linked that box to the transcript. Such linking is a necessary part of good note-taking as it enables the students to reconnect their diagrams to the relevant parts of the arguments. The structural characteristic *FACTS_ISOLATED_FROM_HYPOS* is active when the student has produced a fact node but not linked it to the relevant hypothetical nodes.

These diagram characteristics are associated with phases of graph production (1=orientation, 2=transcript markup, 3=diagram creation, 4=analysis, and 5=reflection). Characteristics of phases 1-3 can be thought of as indicating basic "weaknesses" of a diagram (e.g. *UNLINKED_TEST*), while characteristics of phases 4 and 5 stand for opportunities for reflection contained in a diagram. The system provides feedback on diagrams in the form of self-explanation prompts triggered by the characteristics. In the earlier phases these prompts inform the student about how to fix up the diagrams. In the later phases, the prompts encourage reflection on the diagram and argument representation. These hints are provided upon request.

LARGO also contains a facility for *collaborative feedback*. For each case in the system we have identified two *target* test statements in the transcript. These are test statements that our domain expert considered to be particularly crucial for the analysis process. Students who link a test node to one of these statements are given the opportunity to rate other students' statements

of the same test and to reconsider their own. Students who go through the process and whose tests are rated poorly by their peers are given the opportunity to change their own test in response. This characteristic is *TEST_REVISION_SUGGESTED*. It is active for students whose test has been rated poorly but have not changed the test statement. See [14] for a more detailed analysis of the help system and an argument example.

We have completed three studies of the LARGO system. In the Fall of 2006 we conducted a study with paid volunteers taken from the first year class at the University of Pittsburgh's School of Law (*Novice-2006*). Students were randomly assigned to analyze a pair of cases using LARGO or a text-based notepad tool with no feedback. We compared test scores between the groups and analyzed student interactions with the system. We found no overriding difference between the conditions, and close examination of the questions showed that some were too easy causing a ceiling effect. However on other question types lower aptitude students, as measured by their Law School Admission Test (LSAT) score (a frequently used predictor for success at law schools) in the LARGO condition, showed higher learning gains on some question types than their low-LSAT text peers. Also, the use of the help features was strongly correlated with learning [15].

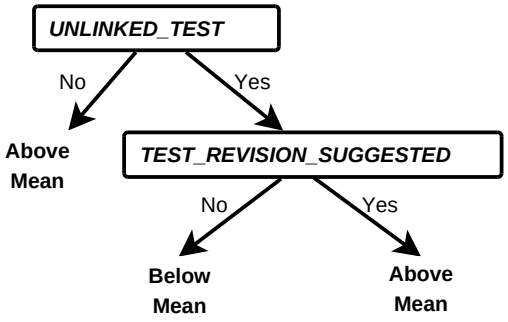
In the Fall of 2007 we performed a follow-up study as part of the first year legal process course (*Novice-2007*). The study was mandatory for all 85 class members. As before students were assigned randomly to text or graph conditions. However the study included one additional case and students answered case-specific essay questions after each session. We also replaced some questions from the pre- and post-tests that had produced a ceiling effect with more challenging alternatives. We again found no significant differences between conditions. A post-hoc analysis revealed that the students in the first study made far more use of the advice functions than the students in the second study, which may explain the difference between the study outcomes.

We are presently conducting a follow-up study with LARGO among third-year law students (*Expert-2008*). All participants in this study used LARGO and performed the same set of tasks as those in *Expert-2007*. The purpose of this study is to examine novice-expert differences in the context of LARGO. At the time of this paper a total of 17 third-year students have completed the study and their data, along with data from the Novice-2007 study, are employed below.

C4.5 is a decision tree induction algorithm [16]. When presented with data it induces an n-ary decision tree that acts as a functional classifier. Each interior node of the tree represents a logical test that branches to one child or another based upon the outcome of the test. Leaf nodes represent predictions or decisions made. Decision trees are traversed from root to leaf. Each decision path p from the root to a leaf node defines a class of cases based upon the relevant features and a classification tag to be assigned to those cases. Each tree therefore represents a hypothesis or test for carving up the space of diagrams according to the factors involved in each.

One such decision tree, and its representation in pseudocode are shown in Figure 2. As we shall discuss below this tree predicts student scores as at or below the mean (0) or above the mean (1) where the inner nodes represent tests for the presence or absence of graph characteristics and the leaves, classifications. Each unique path from root to leaf in a decision tree defines a distinct class of objects. For our purposes these denote unique classes of student solutions.

Intriguingly *TEST_REVISION_SUGGESTED* is taken as a sign of high performance. As we noted above, this is a later phase characteristic and will not be active unless the students have successfully marked up a target region of the diagram with a test node, and then made use of the




```
if (Arg0 has UNLINKED_TEST):
  then:

    if (Arg0 has TEST_REVISION_SUGGESTED):
    then:   return Below_Mean
    else:   return Above_Mean

  else:
     return Above_Mean
```

Figure 2. Sample Decision Tree with Pseudocode.

collaborative feedback with their test summary having been given a poor rating by their peers.

Genetic Programming (GP) is a type of Evolutionary Computation (EC) [12, 2]. In EC algorithms a population of candidate problem solutions is evolved over time via selection mutation, multiple-parent crossover and other operations inspired by biological reproduction. The field of EC arose initially out of work in artificial life [13] and has since been applied in a number of domains including design and decision making.

EC is a stepwise algorithm that starts with a population of randomly generated or externally defined candidate solutions. The fitness of each individual is assessed either by comparison to a gold standard or a competitive "tournament" selection. Based upon this fitness individuals are then permitted to pass their genetic code to the next by means of cloning, combination of genetic material with other fit members, or random mutation. The algorithm as a whole continues until an absolute fitness threshold is reached or a maximum number of generations has passed.

In genetic programming the individual members of the population are interpreted as function code with their performance compared against a target function or task. For the purposes of our experiments the target function was the mapping $\phi : f \rightarrow b_m$ of graph features f to the mean-score bin b_m we discussed above. In this case the raw fitness is defined by the ratio of correct classifications to total classifications in the set.

One disadvantage of GP is the tendency of systems using unconstrained representations (like ours) to select for code that is not just successful but genetically *robust*. Such code is characterized by *introns* or redundant code elements that protect the core function from destructive crossover and slow the discovery of new solutions. This necessitates the use of *parsimony pressure* to control code growth. In this project we applied a scaled penalty based upon size. Higher performing trees were assessed a larger penalty than lower performers.

In this experiment we made use of two reproduction operators: mutation and crossover. Under mutation an individual is copied directly into the next generation with a sub-tree being replaced by a new, randomly generated sub-tree. Under crossover, two parents exchange randomly selected sub-trees with the children being passed to the next generation.

Members of the population at time t are selected for reproduction based upon their fitness. Some forms of GP select individuals proportionally according to their absolute fitness. However this often results in extreme genetic drift toward initially fit individuals and reduces the selection pressure as the $\sigma_{f,t}$ goes down. We therefore employed sigma scaling to assign each individual a

reproductive fitness value of:

$$Exp(i,t) = \begin{cases} 1 + \frac{f_i - \bar{f}_t}{2\sigma_t} & : \quad \sigma_t \neq 0 \\ 1.0 & : \quad \sigma_t = 0 \end{cases} \tag{1}$$

We then select individuals using Stochastic Universal Sampling [12] which ensures that each individual reproduces at least $\lfloor Exp(i,t) \rfloor$ but no more than $\lceil Exp(i,t) \rceil$ times. Taken together these measures prevent genetic drift by ensuring that selection pressure is still high even as the absolute fitness increases. As a machine learning algorithm, GP has a number of advantages. It is well suited to the evolution of arbitrary structures ranging from neural networks to object-oriented programs. This makes it attractive for our present purposes. However, GP also has a number of disadvantages. As a non-deterministic algorithm it makes fewer guarantees about its performance than a more bounded, biased and specialized algorithm such as C4.5. And, while the statistical behavior of the system and use of proper tuning work to prevent random drift, it cannot be completely eliminated. It is also computationally costly; each of our runs required 12 hours of operation on a modern PC.

3 Results and Analysis

As stated in the introduction, our goal in this study is to examine the prospects for automatically inducing higher-order pedagogical knowledge from subject graphs. By analyzing subject graphs using machine learning methods we seek to identify potential target rules for the classification of successful and unsuccessful learners and to explore the interaction of the graph characteristics.

For purposes of this analysis we made use of the final graphs and post-test scores taken from the Novice-2007 and Expert-2008 studies. In both studies the subjects followed the same procedure and took the same tests. The graphs we analyzed were produced for two competing arguments in the Burnham v. Superior Court (495 U.S. 604) case, each of which was represented as a single unified set of graph characteristics as interpreted by LARGO. The post-test score was a single value representing their overall score in the absolute range. We elected to use the final graph students generated in the course of the study as it was created as the culmination of the students' training and thus, was most likely to be correlated with their final performance.

We were forced to remove some of the Novice-2007 subjects from our analysis as they took too little time on the post-test or too little time to read the cases (both indicating a lack of serious effort) or, in the case of four, because they candidly informed us that they were not trying to answer the questions. This left us with 34 students from the Novice-2007 study and 17 from the Expert-2008 study giving us a total of 51 graph/test pairs.

We binned the graph/test pairs according to their post-test score. We then binned the subjects by mean score (0.63) into two groups, those above the mean, and those at or below it. Of the 51 students 22 were below the mean while 29 were above it. Two of the Expert-2008 subjects fell below the mean score. Our χ^2, C4.5 and GP analyses below are based upon this grouping.

Statistical Comparisons: As we report in [10], simple statistical analyses of the graph features such as the number of nodes or relations do not correlate highly with the students' learning outcomes. This was true both for the full set of 51 subjects as well as the study subgroups. While some of the measures *do* correlate with group membership (i.e., expert students produce more interconnected graphs than non-experts) they do not correlate with students' ultimate performance.

Our analysis showed no overall correlation between the phase groups and student performance. Again while there was some difference between the study groups those differences were not significant. However, once we binned the full set of graph results by mean score a distinction emerged. In particular two of the characteristics were significantly correlated with bin membership. *UNLINKED_HYPO* was significantly correlated with having a less than average score ($c^2(16.16, N = 51) = 1.00, p < 0.001$) as was *UNLINKED_TEST* ($c^2(18.27, N = 51) = 1.00, p < 0.001$). This highlights the importance of students' linking of tests and hypos in their diagrams to the argument transcript. In addition, *TEST_REVISION_SUGGESTED* was marginally significantly correlated with high performance ($c^2(4.07, N = 51) = 1.00, p < 0.05$). As the reader will recall this is a 'late phase' characteristic and requires some successful problem solving steps to occur before it is active. This strong correlation of the linking features with student performance fits our domain model. The connection of diagram elements to the argument transcript enables students to retain the context for each note and helps them to develop a "respect for the text" that is a goal of legal instruction.

The significance of *TEST_REVISION_SUGGESTED* prompted us to examine the student help behavior more closely. As you will recall this characteristic is activated when a student has marked up the target region, completed collaborative feedback, but not changed his test. From that we determined that very few of the students modified their test statements in response to this characteristic. Thus students who reached this point are not sufficiently differentiated. This led us to conclude that additional effort must be made to motivate student's help usage, particularly in the later "reflective" phases of work.

C4.5: We split the individual datapoints evenly into a 90/10 Test-train split with 45 Training cases (19 at or below the mean and 26 above) and 6 test cases (3 at or below and 3 above). We did not perform an iterative cross-validation as our goal was to induce information from known algorithms, not to validate the algorithms on existing data. C4.5 produces the pruned tree that is shown in Figure 2. This tree successfully classifies 82.2% of the training cases and 100% of the test cases. Interestingly the only graph features employed within it are *UNLINKED_TEST* and *TEST_REVISION_SUGGESTED*.

In many respects the tree supports the χ^2 analysis in highlighting the importance of the transcript linking, particularly for test nodes. Students who link their nodes to the transcripts do well while those who do not are split between students who receive a *TEST_REVISION_SUGGESTED* and are above the mean and those who are not. Thus students who perform well in other respects by highlighting the key transcript region, summarizing it, and partially completing collaborative filtering have been able to avoid linking all of their test nodes.

GP: For the Genetic Programming experiments we employed the same test/train split over all as with C4.5. On both the Mean classification task the evolutionary algorithm showed early successes. As of generation 93 the system produced the Mean classification tree shown in Figure 3. This tree correctly classified 87% of training cases successfully and 100% of test cases. Subsequent generations showed some improvements with the system achieving 89% correct classification of training instances as of generation 659. However the resulting trees were quite large suggesting a problem of overfitting the data. Introns were already present at generation 93 as shown by the useless appearances of *NO_ELEMENTS* and *ISOLATED_HYPO_DISCUSS* and the frequency of such code only increased as the process went on. Note also that both the *UNLINKED_TEST* and

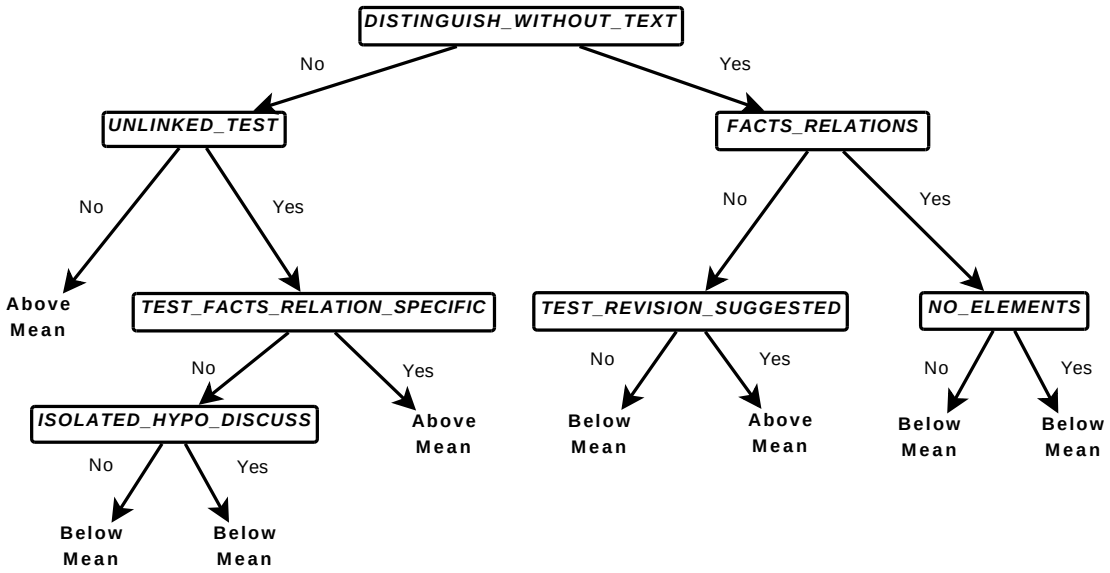


Figure 3. GP Mean Decision Tree.

TEST_REVISION_SUGGESTED rules are present in this tree but not *UNLINKED_HYPO*.

In analyzing the tree shown in Figure 3 we will focus on the three classes of poor performing students that it defines. The root node of the tree is *DISTINGUISH_WITHOUT_TEXT*. This characteristic is active when a student has noted that one node is distinguished from another (irrespective of node type) without giving a justification. As we noted above distinctions, according to our model, are always motivated by some principled or factual justification with which the student should annotate the arc.

As with the C4.5 tree students who exhibit this characteristic alone are rated low unless they also exhibit *TEST_REVISION_SUGGESTED*. The significance of this characteristics prompted us to examine the student help behavior more closely. Recall that this characteristic is activated when a student has marked up the target region, completed collaborative feedback, but not changed his test. From that we determined that very few of the students modified their test statements in response to this characteristic.

Students who exhibit both *DISTINGUISH_WITHOUT_TEXT* and the *FACTS_RELATIONS* characteristic are classified as below the mean. This latter characteristic indicates that the students added arcs relating fact nodes to one another. Again this is a violation of our model. Taken together these characteristics indicate a misunderstanding of the role of facts in the domain model both in terms of how distinctions are drawn using facts and how nodes may be interrelated.

The third class is the set of students who do not exhibit *DISTINGUISH_WITHOUT_TEXT* but do exhibit *UNLINKED_TEST* and *TEST_FACTS_RELATION_SPECIFIC*. This latter characteristic is active when the student has constructed a specific relation (e.g., “Modified To”) to the facts of the case. This is an example of the system’s providing more novel pedagogical information. While our domain model endorses the use of general relationships between the test and fact nodes, it is clear that some good students, who leave tests unlinked, also choose to use specific relations for test and fact nodes. This may signal a valid alternative to our model that bears further exploration.

4 Conclusions

In this paper we assessed the potential of inspectable machine learning methods to induce useful domain information from student work. Our goal was to demonstrate the potential of these methods to yield useful insights into the quality of student solutions, the tutoring system's behavior, and the domain itself. The results we describe above have led us to conclude that these methods do hold potential for domain exploration but not without some measure of guidance.

Our statistical analysis highlighted three salient graph characteristics one of which demonstrated how the use of the system by well-performing students was at odds with our desires. However apart from that it validated our domain model. The use of C4.5 further confirmed the above results and helped to validate some of our domain assumptions but otherwise did not yield much in the way of new information. GP by contrast yielded a set of classification trees one of which we presented here. Examination of this tree yielded useful information both about student misconceptions and student divergence from our domain model. This information has led us to consider alterations to the advice system and a reconsideration of some aspects of our domain model.

These results lead us to conclude that the use of GP to induce "inspectable" classifiers is a fruitful method of data extraction both for behavioral and pedagogical information. We believe that this process is especially useful in ill-defined domains where the relationship among individually detectable solution characteristics is not clear and the means for assessing them, open for debate. In cases such as these the use of inspectable post-hoc classification has been shown to reveal useful insights.

We plan to expand upon this work by moving from the present high-level graph classification into the induction of both lower level graph characteristics and student classifiers that track performance over the course of the study, again with the goal of identifying useful pedagogical and performance information. At the same time we plan to combine these automatic insights with expert human grading. This summer we will engage the services of law school instructors to grade the student graphs. We will then use this data to compare our assessment of good and poor students with theirs and make use of their data to train further classifiers. This will enable us to check the value of our present grading mechanism and to provide expert-level analysis to augment our existing classifications.

Bibliography

[1] Ryan S.J.d. Baker, Albert T. Corbett, Ido Roll, and Kenneth R. Koedinger. Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction.* (to appear).

[2] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming; an Introduction.* Morgan Kaufmann Publishers; San Francisco, 1998.

[3] Hao Cen, Kenneth R. Koedinger, and Brian Junker. Is over practice necessary? - improving learning efficiency with the cognitive tutor through educational data mining. In Luckin et al. [8], pages 511–518.

[4] Matthew W. Easterday, Vincent Aleven, and Richard Scheines. 'tis better to construct than to receive? the effects of diagram tools on causal reasoning. In Luckin et al. [8], pages 93–100.

[5] Mingyu Feng, Niel T. Heffernan, and Kenneth R. Koedinger. Predicting state test scores better with intelligent tutoring systems: Developing metrics to measure assistance required. In Ikeda et al. [7], pages 31–40.

[6] Andreas Harrer, Rakheli Hever, and Sabrina Ziebarth. Empowering researchers to detect interaction patterns in e-collaboration. In Luckin et al. [8], pages 503–510.

[7] Mitsuru Ikeda, Kevin Ashley, and Tak Wai Chan, editors. *Proc. of the 8th International Conference on Intelligent Tutoring Systems. Jhongli Taiwan*. Springer-Verlag Berlin, 2006.

[8] Rosemary Luckin, Kenneth R. Koedinger, and Jim Greer, editors. *Proc. of the 13th International Conference on Artificial Intelligence in Education (AIED2007)*. Amsterdam (Niederlande), IOS Press., 2007.

[9] Collin Lynch, Kevin Ashley, Vincent Aleven, and Niels Pinkwart. Defining ill-defined domains; a literature survey. In Vincent Aleven, Kevin Ashley, Collin Lynch, and Niels Pinkwart, editors, *Proc. of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th International Conference on Intelligent Tutoring Systems. Jhongli (Taiwan), National Central University.*, pages 1–10, 2006.

[10] Collin Lynch, Niels Pinkwart, Kevin Ashley, and Vincent Aleven. What do argument diagrams tell us about students' aptitude or experience? a statistical analysis in an ill-defined domain. In Vincent Aleven, Kevin Ashley, Collin Lynch, and Niels Pinkwart, editors, *Proc. of the Third International Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 9th International Conference on Intelligent Tutoring Systems. UQUAM Montreal, Canada.*, pages 1–9, 2008. (to appear).

[11] Jan Miksatko and Bruce M. McLaren. What's in a cluster? automatically detecting interesting interactions in student e-discussions. In *Proc. of the 9th International Conference on Intelligent Tutoring Systems. UQUAM Montreal, Canada.*, 2008. (to appear).

[12] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge Massachusetts, 1999.

[13] Melanie Mitchell and Stephanie Forrest. Genetic algorithms and artificial life. In Christopher G. Langton, editor, *Artificial Life; An Overview*, pages 267–289. MIT Press, Cambridge Massachusetts, 1999.

[14] Niels Pinkwart, Vincent Aleven, Kevin Ashley, and Collin Lynch. Toward legal argument instruction with graph grammars and collaborative filtering techniques. In Ikeda et al. [7], pages 227–236.

[15] Niels Pinkwart, Vincent Aleven, Kevin Ashley, and Collin Lynch. Evaluating legal argument instruction with graphical representations using largo. In Luckin et al. [8], pages 101–108.

[16] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers; San Francisco, 1993.

The Composition Effect: Conjunctive or Compensatory? An Analysis of Multi-Skill Math Questions in ITS

Zachary A. Pardos – Joseph E. Beck – Carolina Ruiz – Neil T. Heffernan
Worcester Polytechnic Institute

zpardos@wpi.edu – joseph.beck@educationaldatamining.org – ruiz@cs.wpi.edu – nth@wpi.edu

Abstract. Multi skill scenarios are common place in real world problems and Intelligent Tutoring System questions alike, however, system designers have often relied on *ad-hoc* methods for modeling the composition of multiple skills. There are two common approaches to determining the probability of correct for a multi skill question: a conjunctive approach, which assumes that all skills must be known or a compensatory approach which assumes that the strength of one skill can compensate for the weakness of another skill. We compare the conjunctive model to a learned compositional function and find that the learned function quite nearly converges to the conjunctive function. We can confidently report that system designers can implement the AND gate to represent the composition function quite accurately. Cognitive modelers may be interested in the small compensatory effect that is present. We use a static Bayesian network to model the two hypotheses and use the expectation-maximization algorithm to learn the parameters of the models.

1 Introduction

Real world problems often involve multiple skills. It is common to have a question in an Intelligent Tutoring System (ITS) that involves more than one skill as well. It is assumed that multi skill questions are harder but how much harder and is the composition of skills accurately modeled by an AND gate or is a more complex function required?

There are two general approaches that have been used in ITS to model how skills compose; one is a conjunctive (AND) approach which multiplies the probabilities that the student knows the skills together to deduce the probability that the student has the required knowledge to answer the question correctly. One example is the Andes physics tutor [4] which uses a leaky-AND assumption to model composition. The other approach is compensatory (OR/additive). This approach assumes that a skill with high probability of knowledge can compensate for a lower probability of knowledge of the other skill(s) involved and thus constitute the required knowledge to answer the question correct. Many varieties of the compensatory approach have been applied in practice including a mixed-effect analysis of tutor data [5] where skills with the lowest probability were used to represent the problem's difficulty. Other investigators have used a type of additive approach with learning factors analysis [3] and with a linear logistic test model [1] that found the additive approach to be inadequate for modeling difficulty using a similar dataset to ours. This paper will investigate how difficulty changes with the number of required skills and address which function is most appropriate for modeling the composition of skills at least in the context of our dataset. Previous work [6], in which the phrase "composition effect" was first

coined, focused on a different aspect of problem difficulty and found that answering a multi skill question correctly was harder than answering two single skill questions correctly.

1.1 About the ASSISTment tutoring system

ASSISTment is a web based tutoring and assessment system for 6^{th}-10^{th} grade math. The tutor started in 2004 in only a few 8^{th} grade math classrooms and, in 2008, is now used by 300-500 students per day. The items in the tutor are all based upon publically released state test problems from the Massachusetts Comprehensive Assessment System (MCAS).

1.2 About the Dataset

Our dataset was from logged student use of the ASSISTment system during the 2005-2006 school year in which 8^{th} grade math students ages 13-14 answered questions on the tutor two or three times per month at their school's computer lab. The students were given randomly chosen problems from a pool of around 300. The random selection of problems gave us a random sampling of skill data throughout the year.

Our dataset contained 441 students with 46 average responses per student. Only a student's first response to a question was considered. The number of single skill questions was 184 with 62 double skill questions and 16 triple skill questions. The majority of questions are text entry while the others are multiple-choice. We would like to note that skills were tagged to questions by subject matter experts from a skill set of 106 [10]. However, the number of skills that have data points is 84. Table 1 shows the attributes we have for each of the question responses and an example of what the response data looks like.

Table 1. Sample of the question response dataset

UserID	QuestionID	Skill Tags	Correct on first attempt	Timestamp
1234	5293	Equation-solving, Congruence, Perimeter	0	13-OCT-05 12:20.53
1234	5693	Congruence	1	13-OCT-05 12:27.20
1234	6755	Pythagorean Theorem	0	13-OCT-05 12:33.57

1.3 Bayesian Networks

We used a static Bayesian model to represent skill tagging to question nodes and to make inferences on the probability of answering a given question correct or incorrect. Bayesian networks [9] is a powerful machine learning method that we used for making posterior inferences based on binary response data; 1 for correct

0 for incorrect. The EM algorithm [2] can be used with Bayesian networks to learn the priors of latent variables and the conditional probability tables (CPT) of random variable child nodes. Exact inference with Kevin Murphy's Bayes Net Toolkit for MATLAB was used with EM to learn the parameters of the networks.

Inferring the probability a student will answer a question correctly (Q=correct) is a function of the believed prior on the skill(s) associates with the item (S=known) together with the guess and slip parameters of the question. The equation is shown below:

$$P(Q=correct) = (1-P(S=know)) \cdot guess + P(S=know) \cdot (1-slip) \quad (1)$$

A guess parameter dictates the probability that the student will get an item correct even if she does not have the required knowledge. A slip parameter dictates the probability that the student will get an item incorrect even if she has the required knowledge. Learning the general parameters of single and multi skill questions will tell us if questions with more skills are harder to guess.

2 The Conjunctive Model

The AND Gate Conjunctive model is the most common approach to skill composition in ITS. The principle behind it is that all skills involved must be known in order to answer the question correctly. The topology of this model is similar to a deterministic input noisy "AND" (DINA) model [7] except that our AND has no noise (no p-fail). The Bayesian belief network is represented by the directed acyclic graph in Figure 2.

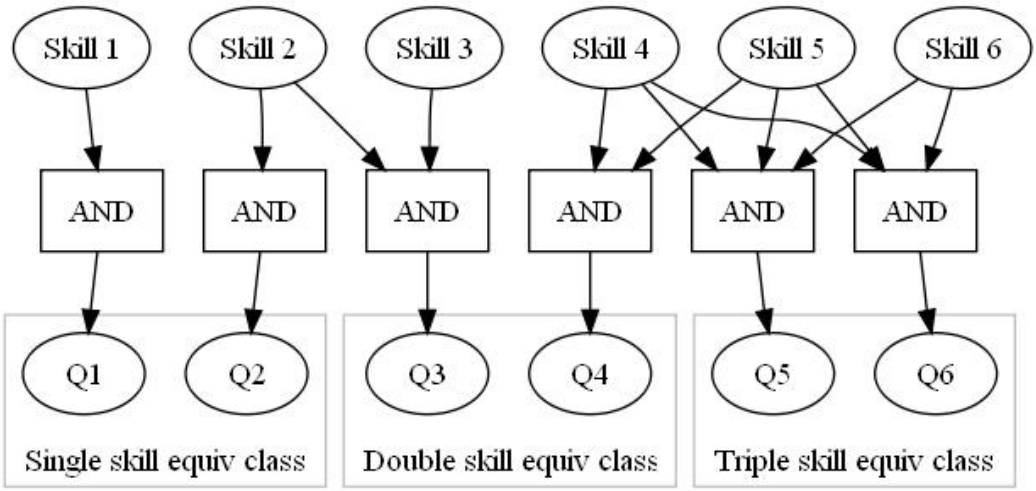


Fig. 2 Directed acyclic graph representing the AND gate Bayesian topology

The network consists of three layers of nodes with equivalence classes used to share conditional probability tables among nodes. This lets us learn a generalized

guess/slip for all nodes of a given equivalence class instead of a parameter per node, which could not be accurately learned given the size of our dataset and would also not answer the research question of how multi skill questions differ from single skill questions. The first layer consists of latent skill nodes. All the skill nodes share a single equivalence class, this was done to simplify the EM procedure. The equivalence class learns a single prior for all skills but does not constrain the individual skill performance estimations from differing. The second layer consists of the AND gates which assert that all parent skills must be known in order for the child question to be answered correctly. The last layer consists of the question nodes. All single skill questions are grouped by an equivalence class. All double skill and triple skill questions have their own equivalence class as well. We will eventually be learning a total of three sets of guess/slip values and a prior.

2.1 Methodology

The way in which we will approach the research question "how much harder are multi skill questions" is by learning the conditional probability tables of the Bayesian network. By learning the parameters of our model we can observe how skill knowledge determines the probability of correct for a given question. How guess and slip vary with the number of skills involved is one way of investigating the composition effect. To learn parameters in the AND model we chose to learn the single skill guess and slip first. By learning these parameters first we can establish a baseline assumption for question guess and slip that is gathered from single skill question data that does not introduce the complex issue of credit-blame that comes in to effect with multi skill questions. The skills' prior equiv class was set at 0.50 and the starting value of the single skill equiv class was set at *ad-hoc* values; 0.15 guess and 0.10 slip that have been used in previous conjunctive model work [8]. After the guess and slip parameters have been learned for the single skill equivalence class we move on to learning the double skill and triple skill equivalence classes at once. The single skill parameter values are locked in place and the prior is reset again to 0.50 before the second EM learning begins. After this second step completes we now have three sets of guess and slip values as well as a prior for the skills.

2.2 Results

The results from the AND model parameter learning shows that the probability of guess decreases linearly as the number of skills increases; from a 24.24% guess with a single skill questions down to a 16.06% guess with a 3 skill question as shown in Figure 3. Surprisingly the slip rate, or probability of making a mistake, also goes down as the number of skills increase. This suggests that while multi skill problems are more difficult to guess, they are also more difficult to slip on.

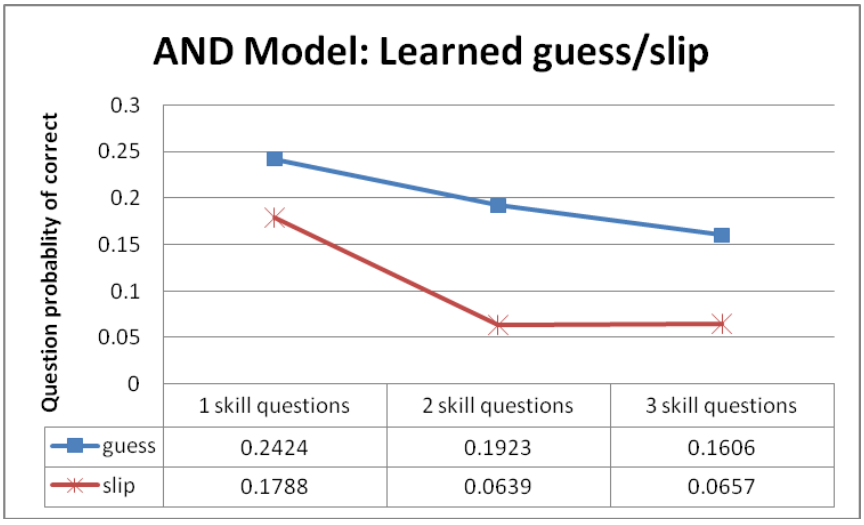


Fig. 3 Results of AND model parameter learning

The difficulty of a problem can be described by the probability of answering the question correctly. However, the probability of answering a question correctly is dependent on the probability of knowing the skill or skills involved. Figure 4 shows how problem difficulty differs with skill knowledge for single, double and triple skill questions. Also note that the guess values from Figure 3 are the intercepts of the left most vertical axis in Figure 4 and the slip values are the right most vertical intercepts.

Data for the Figure 4 graph was generated by setting the probability of all skill nodes to zero and then asking the Bayes net to infer the posterior probability of correct for a single, double and triple skill question and recording the results. All skill nodes were then incremented by 0.01 and the steps were repeated up to a probability of 1.

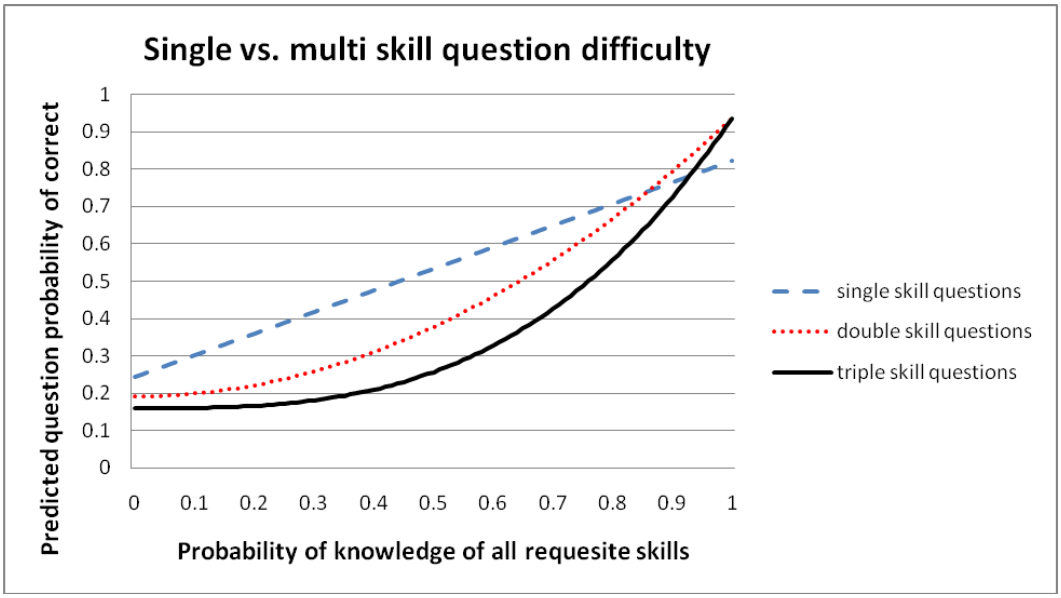


Fig. 4 Comparison of the difficulty of single, double and triple skill questions

3 The Learned Compositional Model

The learned compositional model topology looks similar to the AND model except that there is no layer of gate nodes and the skills are connected directly to the question nodes as seen in Figure 5.

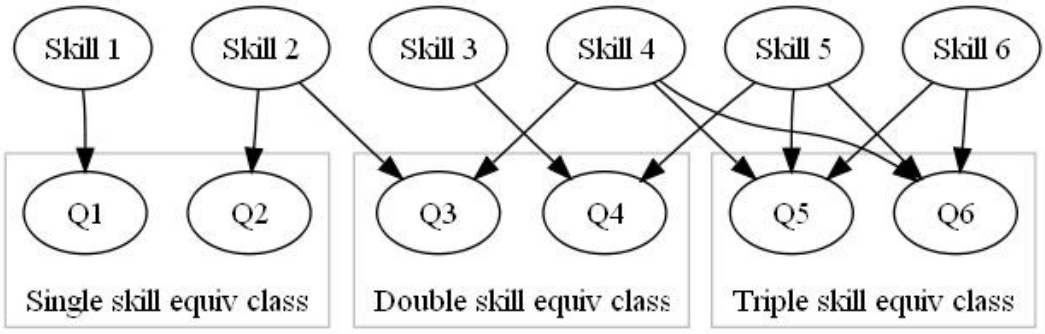


Fig. 5 Directed acyclic graph representing the compositional Bayesian topology

The fundamental difference between the AND model and the learned compositional mode is that only three parameters, a guess and slip and prior, are learned in the AND model since the composition function was captured by the AND gates + guess/slip. In the learned compositional model, however, the composition function and the guess/slip parameters are captured in one complex CPT. A guess and slip value will still be learned for the single skill equivalence class since there is no composition with a single skill. However, four parameters will be learned for the double skill equivalence class and eight parameters for the triple skill class. The increase in parameters is due to the CPT needing to expand with the increased number of parent nodes. Example CPTs for single and double skill questions are shown in Figure 6.

S1	P(Q=F)	P(Q=T)
F	0.85	0.15
T	0.10	0.90

S1	S2	P(Q=F)	P(Q=T)
F	**F**	0.85	0.15
T	**F**	0.80	0.20
F	**T**	0.78	0.22
T	**T**	0.10	0.90

Fig. 6 Example CPTs for single and double skill questions

In both CPTs of Figure 6 the 0.15 represents the guess parameter which is the probability the question is answered correctly (P(Q=T)) given the skill is not known (S1=F); 0.85 is simply the complement of the guess parameter. Observe that in the double skill CPT there is a row where S1=T and S2=F and another row where S1=F and S2=T. Why might the P(Q=T) of these two rows differ? Because, for example, S1 could represent the "harder skill". If the composition function is compensatory, knowledge of only the more difficult skill could results in a P(Q=T) of greater than 0.50 while knowledge of only the easier skill could result

in a P(Q=T) of less than 0.50. In the next section we describe how the network topology was organized to capture the notion of a "harder" skill. We found that knowing the harder skill is not much better than knowing the easier skill, suggesting against a compensatory compositional function.

3.1 Methodology

The methodology for learning parameters in the learned compositional model was very similar to the AND model with the exception of an initial step required to order the skill nodes. This ordering was done to capture relative difficulty among skills so that a compensatory function, which requires the notion of skill difficulty, could potentially be learned. The default order of skills in the network was alphabetical. Because this ordering has no substantive meaning we decided to order the skills by difficulty with the most difficult skill appearing first (node 1) and the least difficult appearing last (node 106). We used the metric of skill prior to represent the difficulty of a skill. In order to attain the priors on all the skills we let a separate prior be learned for each of the 106 skills during an initial parameter learning of single skill questions. This step was done solely to get a skill order. The learned guess/slip values were discarded. After the skill priors were attained, the network was reordered and the single equivalence class for skill priors was reestablished before the "real" first phase of EM parameter learning was run. This reordering gave extra power to the results and allowed us to ask composition questions such as, "does knowing only the harder skill increase the probability of answering a question correctly over knowing only the easier skill?"

3.2 Results

Results of the compositional model parameter learning indicate that the learned composition function is conjunctive. Evidence against a compensatory function can be drawn from Table 2 which shows that knowing one skill only slightly increases the probability of answering a double skill question correctly over knowing no skills.

Table 2. Learned double skill CPT for the learned compositional model

Harder skill	Easier skill	P(Q=F)	P(Q=T)
F	**F**	0.83	0.17
T	**F**	0.77	0.23
F	**T**	0.78	0.22
T	**T**	0.06	0.94

The table also shows that knowing only the harder skill does not help significantly over only knowing the weak skill. For double skill questions the learned guess is 0.17 and 0.06 slip, nearly the same as the AND model. To further compare and verify the learned compositional model's similarity to the AND model we generated a graph similar to Figure 4.

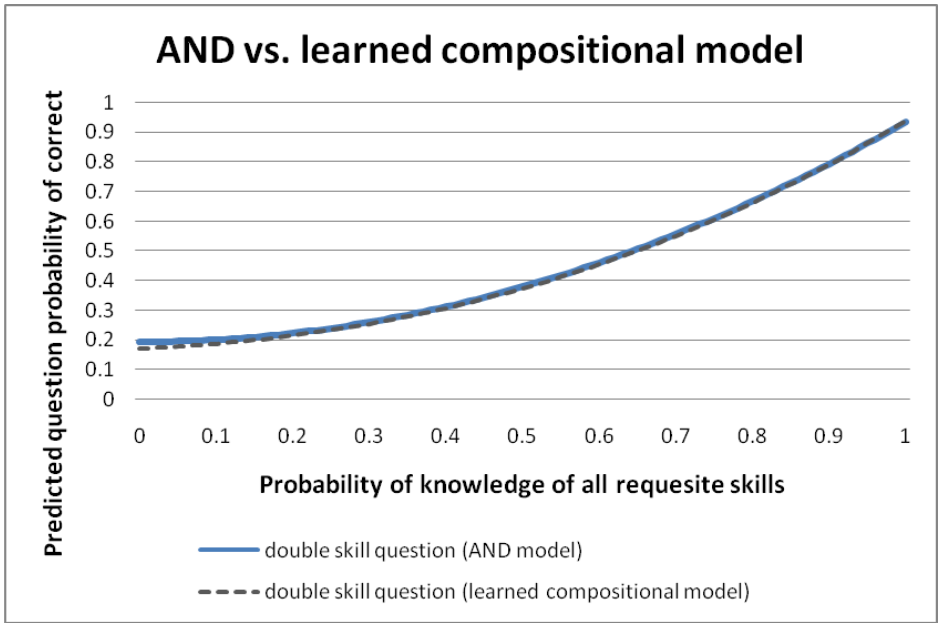


Fig. 7 Comparison of the AND model and learned compositional model

Figure 7, above, shows that the lines from the AND model and learned compositional model overlap. The similarity of the behavior of these functions, arrived at through two different analytic approaches, favors the AND gate as being a very close approximation to the composition function.

3.3 Further analysis: deriving the composition function

The values in Table 2, estimated by our model, determine how likely a student is to respond correctly to a multi-skill question on the basis of her knowledge of the two associated skills. For example, a student who knows the harder skill but does not know the easier skill has a 23% chance of responding correctly to the question. These parameters values are affected by two components: the effect of composition and the likelihood of slipping or guessing. Unfortunately, it is impossible to simultaneously model both effects separately since the model is underdetermined. Both the guess/slip and composition effects are latent, and therefore during the parameter estimation phase of EM there are an infinite number of simultaneous solutions. Therefore, we will reuse the slip and guess values from the AND model (guess=0.1923 and slip=0.0639, from Figure 3) as estimates of the effect of slipping and guessing. We then partial those out (analogous to a partial correlation) of the table by using the following equation: P(correct) = (1-P(known)) * guess + P(known) * (1-slip). In this formula, P(known) refers to the probability the student knows how to solve the problem accounting for the composition effect; P(correct) is the value in Table 2, and slip and guess are 0.1923 and 0.0639, respectively. By solving for P(known), we can compute the effect of composition: that is, when a student knows neither, one or both of the two skills, how much effective knowledge does she bring to bear on the problem? Solving for P(known) for each entry in Table 2 yields Table 3.

Table 3. Compensatory model with guess/slip factored out

P(known)	Hard skill is not known	Hard skill is known
Easy skill not known	-0.028	0.055
Easy skill is known	0.045	1.004

Table 3 represents the composition function. Although some of the values are impossible in terms of being probabilities, they are only (close) approximations since we were forced to use slip and guess values from a related analysis. In spite of these shortcomings, Table 3 is quite interpretable: it is extremely similar to an AND gate. When a student knows neither skill she has, effectively, zero knowledge. When she knows both skills her knowledge is approximately 1. When she only knows 1 of the skills, she has a slight amount of knowledge, but still fairly close to 0. Replacing these numbers with an AND gate would produce largely similar results, as can be seen in Figures 4, 5 and 6. Therefore, composition is well-modeled as an AND gate. Furthermore, we see no evidence that it is necessary to use a leaky-AND gate [4] to model composition.

4 Contributions

We have shown that the more skills involved in a question the harder it is to guess the correct answer. We have also shown that the probability of slipping goes down as well with more skills. We speculate that the reason for decreased slip is that students who are believed to know multiple skills are less likely to make a mistake. Another possibility is that multi skill questions demand more concentration and thus a student is less likely to make a careless mistake due to not paying attention. We also found that knowing the more difficult skill in a multi skill question does not help much over knowing only the less difficult skill or over knowing neither skill.

We have investigated the composition function and found that it is approximated very well by the AND gate + guess/slip. While cognitive modelers may be interested in the slight compensatory effect seen in Table 3, ITS developers can have confidence in using an AND assumption for accurate assessment when dealing with multi skill items.

Acknowledgements

We would like to thank the Worcester Public Schools and all of the people associated with creating the ASSISTment system listed at www.ASSISTment.org including investigators Kenneth Koedinger and Brian Junker at Carnegie Mellon. We would also like to acknowledge funding from the U.S. Department of Education, the National Science Foundation, the Office of Naval Research and the Spencer Foundation.

References

[1] Ayers E., & Junker B. W. (2006). Do skills combine additively to predict task difficulty in eighth grade mathematics? In Beck, J., Aimeur, E., & Barnes, T. (Eds). *Educational Data Mining: Papers from the AAAI Workshop*. Menlo Park, CA: AAAI Press. pp. 14-20. Technical Report WS-06-05.

[2] Baum, L.E., Petrie, T., Soules, G., & Weiss, N. (1970) A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *The annals of Mathematical Statistics, 41*, 164-171

[3] Cen, H., Koedinger, K., & Junker, B. (2006). *Learning Factors Analysis - A General Method for Cognitive Model Evaluation and Improvement*. Paper presented at the 8th International Conference on Intelligent Tutoring Systems.

[4] Conati, C., Gertner, A., & VanLehn, K. (2002). Using Bayesian networks to manage uncertainly in student modeling. User Modeling & User-Adapted Interaction, 12(4), 371-417.

[5] Feng, M., Heffernan, N. T., Mani, M., & Heffernan, C. (2006). Using Mixed-Effects Modeling to Compare Different Grain-Sized Skill Models. In Beck, J., Aimeur, E., & Barnes, T. (Eds). Educational Data Mining: Papers from the AAAI Workshop. Menlo Park, CA: AAAI Press. pp. 57-66. Technical Report WS-06-05. ISBN 978-1-57735-287-7.

[6] Heffernan, N. T. & Koedinger, K.R. (1997). The composition effect in symbolizing: The role of symbol production vs. text comprehension. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum. pp. 307-312.

[7] Junker, B.W., & Sijtsma, K. (2001). Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement, 25*, 258–272.

[8] Pardos, Z.A., Heffernan, N.T., Anderson, B., Heffernan, C.L. The Effect of Model Granularity of Student Performance Prediction Using Bayesian Networks. In the Complete On-Line Proceedings of the Workshop on Data Mining for User Modeling, at the 11th International Conference on User Modeling (UM 2007) Pages 91-100

[9] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA

[10] Razzaq, L., Heffernan, N., Feng, M., Pardos Z. (2007). Developing Fine-Grained Transfer Models in the ASSISTment System. Journal of Technology, Instruction, Cognition, and Learning, Vol. 5. Number 3. Old City Publishing, Philadelphia, PA. 2007. pp. 289-304.

An open repository and analysis tools for fine-grained, longitudinal learner data

Kenneth R. Koedinger, Kyle Cunningham, Alida Skogsholm, and Brett Leber
{krk, kcunning, alida, bleber}@cs.cmu.edu
Human Computer Interaction Institute, Carnegie Mellon University

Abstract. We introduce an open data repository and set of associated visualization and analysis tools. The Pittsburgh Science of Learning Center's "DataShop" has data from thousands of students deriving 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. Currently over 110 datasets are stored including nearly 18 million student actions. 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 neede to perform that action. Researchers have analyzed these data to better understand student cognitive and affective states and the results have been used to redesign instruction and demonstrably improve student learning.

1 Introduction

One reason for emerging excitement about educational data mining is the increasing availability of student learning data. These data are coming from many sources including data from schools, like standardized tests and student and teacher background variables (e.g., www.icpsr.umich.edu/IAED), and videos of classroom interactions (e.g., www.talkbank.org). In this paper we present an open data repository of learning data coming primarily from computer "click stream" data that arises from student interaction and system response in online courses, online assessment, intelligent tutoring systems, virtual labs, simulations, and other forms of educational technology.

Extensive new data sources have been transformational in science [2] and business (e.g., Google, FedEx). Learning scientists can also benefit from having large repositories of data available, easily accessible, and with associated data visualization and analysis tools. Moving toward a common set of standards for storing data will facilitate more efficient, extensive storage and use of such data, as well as greater sharing of visualization and analysis techniques. Providing learning scientists with rich student learning data and advancing toward common standards are key goals of the Pittsburgh Science of Learning Center's DataShop (http:// learnlab.org/datashop).

2 The Pittsburgh Science of Learning Center's *DataShop*

DataShop is a data repository and web application for learning science researchers. It provides secure data storage as well as an array of analysis and visualization tools available through a web-based interface. Primarily, DataShop stores learner interactions from online course materials that include intelligent tutors. Data is collected from the seven PSLC courses: Algebra, Chemistry, Chinese, English, French, Geometry and

Physics. There are also sources external to the PSLC that contribute to DataShop, such as middle school math data from the Assistments project (http://www.assistment.org).

DataShop can store any type of data associated with a course or study. This includes intelligent tutor data (which is capable of being analyzed through the analysis and visualization tools) as well as any related publications, files, presentations, or electronic artifacts a researcher would like to store. Courses and studies are represented as datasets, which are organized by project. An example of this is the "Algebra 1 2005-2006" dataset, which is grouped with similar datasets under the "Algebra Course" project. Semantic information can be stored for each dataset, providing additional context.

The amount of data in DataShop is constantly growing. As of May 2008, DataShop offers 114 datasets under 42 projects, most generated within the past 4 years. The majority of datasets contain tutor interaction data, while others are files-only datasets, used solely for central and secure file storage. There are about 18 million tutor transactions, representing about 70,000 student hours available for analysis.

Researchers have utilized DataShop to explore learning issues in a variety of educational domains. These include, but are not limited to, collaborative problem solving in Algebra [17], self-explanation in Physics [9], the effectiveness of worked examples and polite language in a Stoichiometry tutor [11] and the optimization of knowledge component learning in Chinese [14].

2.1 Logging & Storage Models

Tutor data is typically parsed from messages logged by automated computer tutors employing the DataShop XML format, such as the one depicted in Figure 1.

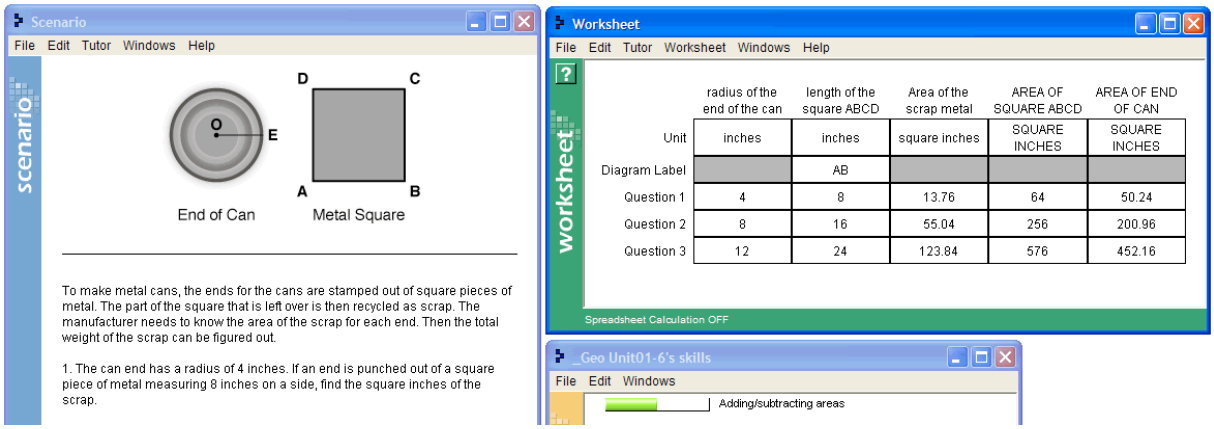


Figure 1. An example problem, "Making Cans", from Carnegie Learning's Cognitive Tutor 2005.

In the "Making Cans" example shown in Figure 1, a student completes a problem where she is asked to find the area of a piece of scrap metal left over after removing a circular area (the end of a can) from a metal square. The student enters everything in the worksheet except for the row labels, and column and 'Unit' labels for the first three

columns. In addition to providing students with feedback and instruction, the tutor records the student's actions and tutor responses, and stores them in a log file, which is imported and analyzed by DataShop.

The DataShop logging format differs from existing standard formats in that it attempts to capture student-tutor interaction history at a fine-grained level, while also providing context for such interactions; the format does not attempt to describe, *a priori*, learning resources and how they're transferred (e.g., LOM, SCORM) or test content (e.g., IMS-QTI). In this way, the format is essentially descriptive, not prescriptive. The DataShop logging model is represented by the following constructs [16]:

- Context message: the student, problem, and session with the tutor
- Tool message: represents an action in the tool performed by a student or tutor
- Tutor message: represents a tutor's response to a student action

Below we see example context, tool, and tutor messages in the DataShop XML format:

```
<context_message context_message_id="C2badca9c5c:-7fe5" name="START_PROBLEM">
  <dataset> <name>Geometry Hampton 2005-2006</name>
    <level type="Lesson"> <name>PACT-AREA</name>
      <level type="Section"> <name>PACT-AREA-6</name>
        <problem> <name>MAKING-CANS</name> </problem>
      </level>
    </level>
  </dataset>
</context_message>
<tool_message context_message_id="C2badca9c5c:-7fe5">
    <semantic_event transaction_id="T2a9c5c:-7fe7" name="ATTEMPT" />
    <event_descriptor>
      <selection>(POG-AREA QUESTION2)</selection>
      <action>INPUT-CELL-VALUE</action>
      <input>200.96</input>
    </event_descriptor>
  </tool_message>
<tutor_message context_message_id="C2badca9c5c:-7fe5">
    <semantic_event transaction_id="T2a9c5c:-7fe7" name="RESULT" />
    <event_descriptor> … [as above] … </event_descriptor>
    <action_evaluation>CORRECT</action_evaluation>
  </tutor_message>
```

In this example, the student attempted problem "MAKING-CANS" in the "PACT-AREA" lesson of the Geometry tutor. Looking at the tool and tutor message pair, we see the student correctly entered "200.96" as the answer. Tool and tutor messages are traditionally paired with each other (by context message), allowing DataShop to interpret the student action and the tutor's response in conjunction. These pairs are then stored as a single *tutor transaction* in the database. Table 1 below illustrates how actions from the Making Cans example are interpreted and stored as tutor transactions.

A tutor transaction stores details such as the student(s), session, time, problem, problem subgoal (step), attempt number, student input, tutor response, number of hints, conditions assigned to the problem step, as well as knowledge components (skills). Unpaired tool messages can be used to represent untutored actions, such as a student starting audio playback, and are stored in the repository as well.

Table 1. A simplified tutor transaction excerpt from the "Making Cans" example.

#	Student	Problem	Step	Attempt #	Student Input	Evaluation	Knowledge component
1	S01	MAKING-CANS	(SQUARE-BASE Q1)	1	8	CORRECT	Enter-Given
2	S01	MAKING-CANS	(SCRAP-METAL-AREA Q1)	1	32	INCORRECT	
3	S01	MAKING-CANS	(SCRAP-METAL-AREA Q1)	2	4	INCORRECT	
4	S01	MAKING-CANS	(SQUARE-AREA Q1)	1	64	CORRECT	Square-Area
5	S01	MAKING-CANS	(POG-AREA Q1)	1	50.24	CORRECT	Circle-Area
6	S01	MAKING-CANS	(SCRAP-METAL-AREA Q1)	3	13.76	CORRECT	Compose-Areas
7	S01	MAKING-CANS	(POG-RADIUS Q2)	1	8	CORRECT	Enter-Given
8	S01	MAKING-CANS	(SQUARE-BASE Q2)	1	16	CORRECT	Enter-Given
9	S01	MAKING-CANS	(SQUARE-AREA Q2)	1	256	CORRECT	Square-Area
10	S01	MAKING-CANS	(POG-AREA Q2)	1	200.96	CORRECT	Circle-Area
11	S01	MAKING-CANS	(SCRAP-METAL-AREA Q2)	1	55.04	CORRECT	Compose-Areas
12	S01	MAKING-CANS	(POG-RADIUS Q3)	1	12	CORRECT	Enter-Given
13	S01	MAKING-CANS	(SQUARE-BASE Q3)	1	24	CORRECT	Enter-Given
14	S01	MAKING-CANS	(SQUARE-AREA Q3)	1	576	CORRECT	Square-Area
15	S01	MAKING-CANS	(POG-AREA Q3)	1	452.16	CORRECT	Circle-Area
16	S01	MAKING-CANS	(SCRAP-METAL-AREA Q3)	1	123.84	CORRECT	Compose-Areas
17	S01	MAKING-CANS	DONE	1	DONE	CORRECT	Determine-Done

Multiple tool and tutor messages are typically logged for a single problem-solving activity. Problem-solving activity is broken down into "steps" which represent completion of possible subgoals or pieces of a problem solution. Students often make multiple attempts at a step or get instructional help on a step and each of these attempts or help requests are stored as a separate tutor transaction in the database.

In the "Making Cans" example, we see the student attempted the "(SCRAP-METAL-AREA Q1)" step three times (transaction numbers 2, 3 and 6 in Table 1). We can ascertain from the transactions that the student was unsuccessful in her first two attempts, providing an answer of "32" and "4", both labeled as incorrect by the tutor. On the third attempt, the student successfully completed the problem step, providing an input of 13.76 (as can be seen in Figure 1).

To allow for fast and easy visualization and analysis of data, tutor transactions are aggregated into a student-step rollup table. This "denormalized" table aggregates steps by student, problem, and step and is used by many of the DataShop tools, such as the Performance Profiler and Learning Curve. An example of how the "Making Cans" tutor transactions are aggregated by student-step is depicted in Table 2.

The student-step roll-up table stores an array of useful step information, some of which is depicted above. In the "(SCRAP-METAL-AREA Q1)" step in the "Making Cans" example, we can see that the three tutor transactions are now rolled into a single step entry (line number 9 in Table 2). "Opportunity count" is two since the student saw the "Compose-Areas" knowledge component twice. "Total incorrects" is two, since she

made two incorrect attempts. "Assistance score", which consists of total hints added to total incorrects, is two. "Error rate", whether the first attempt was an incorrect action or hint request, is 1 (or 100%) because the student's first attempt was an error.

Table 2. Data from the "Making Cans" example, aggregated by student-step

Step table excerpt

#	Student	Problem	Step	Opportunity Count	Total Incorrects	Total Hints	Assistance Score	Error Rate	Knowledge component
1	S01	WATERING_VEGGIES	(WATERED-AREA Q1)	1	0	0	0	0	Circle-Area
2	S01	WATERING_VEGGIES	(TOTAL-GARDEN Q1)	1	2	1	3	1	Rectangle-Area
3	S01	WATERING_VEGGIES	(UNWATERED-AREA Q1)	1	0	0	0	0	Compose-Areas
4	S01	WATERING_VEGGIES	DONE	1	0	0	0	0	Determine-Done
5	S01	MAKING-CANS	(POG-RADIUS Q1)	1	0	0	0	0	Enter-Given
6	S01	MAKING-CANS	(SQUARE-BASE Q1)	1	0	0	0	0	Enter-Given
7	S01	MAKING-CANS	(SQUARE-AREA Q1)	1	0	0	0	0	Square-Area
8	S01	MAKING-CANS	(POG-AREA Q1)	2	0	0	0	0	Circle-Area
9	S01	MAKING-CANS	(SCRAP-METAL-AREA Q1)	2	2	0	2	1	Compose-Areas
10	S01	MAKING-CANS	(POG-RADIUS Q2)	2	0	0	0	0	Enter-Given
11	S01	MAKING-CANS	(SQUARE-BASE Q2)	2	0	0	0	0	Enter-Given
12	S01	MAKING-CANS	(SQUARE-AREA Q2)	2	0	0	0	0	Square-Area
13	S01	MAKING-CANS	(POG-AREA Q2)	3	0	0	0	0	Circle-Area
14	S01	MAKING-CANS	(SCRAP-METAL-AREA Q2)	3	0	0	0	0	Compose-Areas
15	S01	MAKING-CANS	(POG-RADIUS Q3)	3	0	0	0	0	Enter-Given
16	S01	MAKING-CANS	(SQUARE-BASE Q3)	3	0	0	0	0	Enter-Given
17	S01	MAKING-CANS	(SQUARE-AREA Q3)	3	0	0	0	0	Square-Area
18	S01	MAKING-CANS	(POG-AREA Q3)	4	0	0	0	0	Circle-Area
19	S01	MAKING-CANS	(SCRAP-METAL-AREA Q3)	4	0	0	0	0	Compose-Areas
20	S01	MAKING-CANS	DONE	2	0	0	0	0	Determine-Done

Each step in a problem requires the student to know something—a relevant concept or skill—to perform the step correctly. This small unit of knowledge is termed a "knowledge component", a key notion in the PSLC's theoretical framework (learnlab.org/research/wiki). To document this required concept or skill, a tutor author labels steps with a hypothesized knowledge component(s) required for correct completion of the step. In the "Making Cans" example, we see the knowledge component "Compose-Areas" assigned to the correct transaction (row 6 of Table 1) for the "(SCRAP-METAL-AREA Q1)" step.

A knowledge component codes for a general student capability to accomplish steps in tasks. Knowledge component modeling, the process of assigning knowledge components to steps, bolsters the usefulness of intelligent tutor data by providing additional context for a step. A step can have zero, one or multiple knowledge components associated with it. Steps that have assigned knowledge components are used to produce a variety of "learning curves". Figure 2 below depicts error rate learning curves generated by DataShop. In this graph, error rate, or the percentage of students that asked for a hint or made an incorrect attempt on their first attempt on steps associated with the knowledge component, is shown on the y-axis. The x-axis ("Opportunity") indicates the nth time (e.g., 4 is the 4th time) a student has (in theory) used (or tutored on) a knowledge component to solve a step in a problem. Each unique step in a problem is distinct from other problem-solving steps. For example, in Table 2, "(POG-AREA Q1)" and "(POG-AREA Q2)" are unique steps—they correspond to different questions in problem. Because they both exercise the "Circle-Area" knowledge component, they are counted as

distinct opportunities for the student to demonstrate whether he or she has learned "Circle-Area" (and if not, get instruction on it).

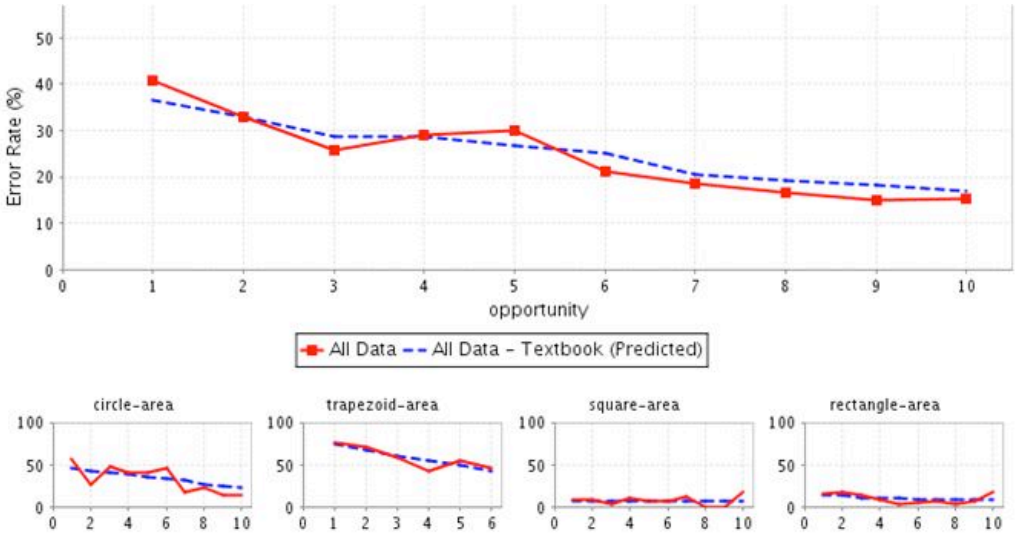


Figure 2. Error Rate Learning Curve with predicted values from a Geometry Area dataset.

The solid curve represents the actual values, each point an average across all students and knowledge components for the given opportunity. The dashed curve represents the predicted curve values, based on the Learning Factor Analysis (LFA) algorithm [5]. Built-in statistical models measure student proficiency, knowledge component difficulty, and knowledge component learning rates. This algorithm allows for search and comparison of alternative knowledge component or cognitive models. DataShop supports users in entering multiple hypothesizes about how student knowledge and learning may be decomposed.

2.2 Importing and Exporting Learning Data

Data may be imported into the DataShop repository through XML or a tab-delimited text file format. Logging to DataShop XML provides the richest and most complete data. If logging via XML, tutors can send messages directly to the DataShop logging server in real time. Logs are automatically processed on a nightly basis, making them available for analysis or export through the web application. Alternatively, a computer tutor can write XML to files on the local hard disk (for example, if the tutor is running off-line) and then send the data to the logging server at a later time. Data in a pre-existing log format can also be converted to DataShop XML and then imported into the repository. This procedure has worked well for data collected by other tutoring systems including Andes (www.andes.pitt.edu), math Cognitive Tutors (carnegielearning.com), REAP (reap.cs.cmu.edu), and Assistments (assistment.org). The tab-delimited format of a transaction table can alternatively be used to import from a preexisting source.

DataShop offers various data export options through the web application each delivered in a tab-delimited text file. These include transaction and student-step level exports (as illustrated in Tables 1 & 2), and a student-problem aggregate export.

2.3 Analysis & Visualization Tools

The DataShop web application provides several tools to assist with analyzing and visualizing repository data. These tools can be used in conjunction to jump-start data analysis: one can determine if students are learning by viewing learning curves, then drill down on individual problems, knowledge components, and students to analyze performance measures.

The following DataShop tools are available:

- *Dataset Info:* provides dataset metrics, contextual information, quick statistics (number of students, transactions, knowledge components, etc.) as well as papers, files, a problem table, and exporting and importing knowledge component models.
- *Performance Profiler:* multi-purpose tool that visualizes student performance across various dataset-specific domains (problem, step, curriculum level, knowledge component and student) and measures of performance (error rate, assistance score, average number of incorrects, average number of hints, and residual error rate).
- *Error Report:* presents each student's first attempt at a problem or knowledge component, including if he or she was correct, the number of students or observations, and the text of the student's answer.
- *Learning Curve:* visualizes student learning changes over time. Learning curve types include error rate, assistance score, correct step time, and others. The Learning Factors Analysis model [6] provides predicted values for error rate learning curves.

3 Uses of PSLC's DataShop

As indicated above, many recent analyses of data from DataShop have been performed in a variety of domains. A number of other studies have used, tested or extended the analysis techniques employed in DataShop including investigations in reading [10], Physics [12], and Geometry [15]. Often analyses have been targeted at finding ways to improve student learning. In some cases, the work has been taken full circle such that an analysis led to an instructional redesign that was demonstrated to improve student learning beyond that realized by the original instruction. We provide a couple examples.

Cen, Junker, and Koedinger [6] performed a learning curve analysis using the Learning Factors Analysis (LFA) algorithm based on data from the Area unit of the Geometry Cognitive Tutor. They noticed that while students were required to over-practice some easy target knowledge components or skills (see square-area in Figure 2), they under-practiced some harder skills (see trapezoid-area in Figure 2). Based on observation and further analysis, they created a new version of the geometry tutor by resetting parameters that determine how often skills are practiced. They ran a classroom experiment where students in a course were pre- and post-tested and randomly assigned to use either the previous or the new tutor version. Students using the new version took 20% less time to finish the same curriculum units (because over-practice was eliminated) and learned just as much as measured by normal, transfer, and long-term retention tests.

A second demonstration of a datamining project that "closed the loop" is work by Baker et al. [3], who had done formal observations of student behavior in computer labs while working through lessons of a middle school math Cognitive Tutor. Among a number of categories of off-task or otherwise disengaged behavior, he found that "gaming the system" had the largest correlation with poor learning outcomes. Gaming refers to student behavior that appears to avoid thinking and learning through systematic guessing or fast and repeated requests for increasing help. Baker used machine learning techniques to build a "detector" capable of processing student log information, in real time, to determine when students were gaming. The detector became the basis for an intervention system, a "meta tutor", designed to discourage gaming and engage students in supplementary instruction on topics they had gamed. A controlled experiment demonstrated student-learning benefits associated with this adaptive selection of supplementary instruction for students observed to be gaming.

4 Discovering better cognitive and affective models of student learning

An important general use of this kind of data is to drive the development of more precise computational models of human cognition, motivation, and learning. The work on gaming is just one example of using interaction log data to assess and remediate student motivation and other detectors of student engagement are possible.

With respect to cognitive and learning assessment, we have been pursuing a strong hypothesis that the correct representation of knowledge (facts, skills, concepts, strategies, integrative models or schemas, meta-knowledge, etc.) in a domain is an empirical question. Why is this a strong claim? First, many do not believe knowledge is decomposable, for instance, Barab and Squire say "learning, cognition, knowing, and context are irreducibly co-constituted and cannot be treated as isolated entities or processes" [4]. Second, most designers of instruction, like textbook authors, assume that the instructional designers determine the knowledge decomposition, the target concepts and skills, and students learn these knowledge components as they are taught. In contrast, the PSLC takes the position that the nature and grain size of mental knowledge representation is driven as much or more by the student and the explicit and implicit learning mechanisms in which they and their brains engage. Others, too, have emphasized that students do not learn knowledge in pieces as they are taught [7]. If the nature of human knowledge representation is an empirical question, then we need both vast and detailed educational data and associated data processing algorithms to answer this question for all domains of academic learning.

In addition to the LFA technique mentioned above, which employs the "smooth learning curve" criteria [13], a number of algorithms have been created for empirically evaluating knowledge representations against student performance data including the rule space [18] and knowledge space [8] approaches. The collection and use of on-line learning data will further drive such developments. As noted in a major national report "psychometric validation of [on-line] assessments is needed so they can be compared with conventional assessments, and complement and ultimately supplant them" [1].

5 Conclusion

We described PSLC's DataShop, an open repository and web-based tool suite for storing and analyzing click-stream data, fine-grained longitudinal data generated by online courses, assessments, intelligent tutoring systems, virtual labs, simulations, and other forms of educational technology. In contrast to other types of educational data such as video and school-level data, data in DataShop includes a rich set of semantic codes that facilitate automated analysis and meaningful interpretation.

The PSLC DataShop uniform data format is an initial attempt to develop a common standard that we hope will be useful to field if not as is, then in driving better or more useful common standards. In addition to being a source for learning data, it is also a place where researchers can deposit data and then get help from other researchers who can perform secondary analysis on this data.

DataShop allows free access to a wide variety of data sets and analysis tools. These tools help researchers visualize student performance, difficulties, and learning over time. Such analyses can lead to demonstrably better instructional designs. The data can also drive improved models of student cognition, affect, and learning that can be used to improve on-line assessment and on-line learning. We take as a premise that the human brain constructs knowledge based a variety of input sources (e.g., verbal, visual, physical) and in a fashion and at a grain size that may or may not conform to the structure as conceived by an instructor or domain expert. In other words, the nature and content of human knowledge representation is a deep and important scientific question, like for instance, the nature of the human genome. To answer this question requires a vast collection of relevant data, associated analysis methods and new theory.

Acknowledgement

Research supported by the National Science Foundation award number SBE-0354420.

References

1. Ainsworth, S., Honey, M., Johnson, W. L., Koedinger, K. R., Muramatsu, B., Pea, R., Recker, M., & Weimar, S. Cyberinfrastructure for Education and Learning for the Future (CELF): A Vision and Research Agenda, 2005. Washington, DC: Computing Research Association. http://www.cra.org/reports/cyberinfrastructure.pdf.
2. Atkins, D.E. (Ed.). Revolutionizing Science and Engineering Through Cyberinfrastructure: Report on the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure, 2003. Arlington, Virginia: National Science Foundation. http://www.cise.nsf.gov/sci/reports/atkins.pdf
3. Baker, R., Corbett, A., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A., Naim, M., Raspat, J., Baker, D., & Beck, J. Adapting to when students game an intelligent tutoring system. In M. Ikeda, K. D. Ashley, T.-W. Chan (Eds.) *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* 2006, p. 392-401. Berlin: Springer-Verlag.

4. Barab, S. & Squire, K. Design-based research: Putting a stake in the ground. *The Journal of the Learning Sciences,* 2004, 13(1).
5. Cen, H., Koedinger, K., & Junker, B. Learning Factors Analysis - A general method for cognitive model evaluation and improvement. *8th International Conference on Intelligent Tutoring Systems,* 2006.
6. Cen, H., Koedinger, K., & Junker, B. Is over practice necessary? – Improving learning efficiency with the cognitive tutor through educational data mining. In Rose Luckin and Ken Koedinger (Eds.) *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, 2007. p. 511-518. Los Angeles: IOS Press.
7. diSessa, A. A. Toward an epistemology of physics. *Cognition and Instruction,* 1993, 10(2 & 3), p. 105-225.
8. Falmagne, J.-C., Koppen, M., Villano, M., Doignon, J.-P., et al. Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review,* 1990, 97, p. 201-224.
9. Hausmann, R., & VanLehn, K. Self-explaining in the classroom: Learning curve evidence. In McNamara & Trafton (Eds.), *Proceedings of the 29th Annual Cognitive Science Society*, 2007, p. 1067-1072. Austin, TX: Cognitive Science Society.
10. Leszczenski, J. M. & Beck J. E. What's in a word? Extending learning factors analysis to model reading transfer. *Proceedings of the Educational Data Mining workshop* held at the 14th International Conference on Artificial Intelligence in Education, Los Angeles, 2007, p. 31-39.
11. McLaren, B. M., Lim, S., Yaron, D., & Koedinger, K. R. Can a polite intelligent tutoring system lead to improved learning outside of the lab? In Luckin & Koedinger (Eds.) *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, 2007, p. 433-440. Los Angeles: IOS Press.
12. Nwaigwe, A., Koedinger, K.R., VanLehn, K., Hausmann, R., & Weinstein, A. Exploring alternative methods for error attribution in learning curves analyses in intelligent tutoring systems. In Luckin & Koedinger (Eds.) *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, 2007. p. 246-253.
13. Ohlsson, S. & Mitrovic, A. Constraint-based knowledge representation for individualized instruction. *Computer Science and Information Systems*, 2006, 1-22.
14. Pavlik Jr., P. I., Presson, N., & Koedinger, K. R. Optimizing knowledge component learning using a dynamic structural model of practice. In R. Lewis & T. Polk (Eds.) *Proceedings of the Eighth International Conference of Cognitive Modeling*, 2007.
15. Rafferty, A. N. & Yudelson, M. Applying learning factors analysis to build stereotypic student models. *Proceedings of 13th International Conference on Artificial Intelligence in Education,* 2007.
16. Ritter, S., & Koedinger, K. R. An architecture for plug-in tutor agents. *Journal of Artificial Intelligence in Education,*1996, 7(3-4), p. 315-347.
17. Rummel, N., Spada, H., & Diziol, D. Evaluating collaborative extensions to the Cognitive Tutor Algebra in an in vivo experiment. Lessons learned. Paper presented at the 12th European Conference for Research on Learning and Instruction (EARLI) 2007. Budapest, Hungary.
18. Tatsuoka, K. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement,* 1983, 20(4), p. 345-354.

Mining Data from an Automated Grading and Testing System by Adding Rich Reporting Capabilities

Anthony Allevato, Matthew Thornton, Stephen H. Edwards, and Manuel A. Pérez-Quiñones
{allevato, thorntom}@vt.edu, {edwards, perez}@cs.vt.edu
Department of Computer Science, Virginia Tech

Abstract. Programs that perform automated assignment grading can generate a great deal of meaningful data not only for the student, but for the instructor as well. Such tools are often used in computer science courses to assess student programming work. In the process of grading, a large amount of intermediate information is gathered. However, in most cases this information is not used beyond assigning scores, so the potential of learning more about the course is lost. At the same time, continuous collection of data over a large number of submissions across many courses presents an interesting but untapped resource for educational data mining. One solution to this problem is to implement a reporting tool for making use of this intermediate data to create meaningful interpretations. This paper describes how an automated grading system, Web-CAT, has been extended to provide a reporting mechanism that uses the intermediate data that is gathered during assessment of students' programs. Our implementation of the reporting tool makes use of the Business Information Reporting Tool (BIRT) developed for the Eclipse IDE.

1 Introduction

In computer science or information technology courses where students learn to write programs, many assignments focus on this skill. How do student grades improve as a function of how close they submit their assignment to the due date? Is there any correlation between the numbers of test cases that a student writes versus the grade they receive? Is there a statistically significant difference in grades from one assignment to the next? These are typical of questions that instructors may ask in analyzing the success of their lesson plans. Instructors that use some form of automated grading or testing system to help assess student work are at a significant advantage. Such systems should be able to help answer these questions because they provide a great deal of feedback to the instructor about student grades and the assessment of individual assignments.

Unfortunately, such applications usually do not provide a useful way of analyzing this information. Web-CAT, an automated testing system [3], collects a large amount of intermediate data when assessing student projects. Prior to our work outlined here, these data were effectively "trapped" within the tool and were not publicly accessible, thus preventing any analysis. Making the data accessible would require an analysis and reporting tool that would allow complex reports to be generated.

The Business Information Reporting Tools (BIRT) project is an open-source reporting engine based on the Eclipse IDE [2]. BIRT makes it possible to take raw data from some data source and render it using tables, crosstabs, and a number of chart types. It can then generate reports in several formats including PDF and HTML. BIRT is also easily extended through the Eclipse plug-in and extension point paradigms.

Integrating BIRT into Web-CAT provides a powerful resource for data analysis for a student, an assignment, or a course overall. Unfortunately, the integration was not as simple as connecting a data store from Web-CAT to BIRT and generating reports, because data from Web-CAT comes from several disparate sources. It was necessary to create a "bridge" between the Web-CAT data store and the BIRT reporting engine. Additionally, tight integration into the existing Web-CAT user interface was also desired.

This paper details our solution to the problem of analyzing the large volume of data that is acquired in assessing student assignments using an automated grading system. We will look at previous work done in analysis of student assessment data, describe the requirements of our system, and discuss the highlights of its design and implementation. Finally, we will show examples of some in-depth analysis that is now possible because of these enhancements and describe future directions for this tool and its potential to aid in student, assignment, and course assessment from the instructor's perspective.

2 Background and Related Work

Web-CAT is a web-based system that supports the electronic submission and automated grading of programming assignments [3]. In addition to traditional models of automated grading, it also supports test-driven development by evaluating the correctness and thoroughness of unit tests that students submit with their solution. When students submit, they receive a report detailing their performance on the assignment. This report includes a score derived from such sources as static analysis and style-checking tools, code coverage monitoring tools, and the results of executing reference tests that the instructor provides. Students can also view visualizations of their past performance and their standing among other students [5]. However, a great deal of the raw data that is used to derive these final scores is left inaccessible, buried in internal databases and other file structures. Much of this data could potentially be of use to an instructor who wishes to assess either the abilities of students based on other criteria than their final cumulative scores, or the effectiveness of an assignment by examining how students at different skill and knowledge levels were able to perform on a particular aspect of the assignment.

Marmoset, developed by Spacco, Strecker, et al., is a system that performs analysis of student submissions beyond the computation of a final score for grading purposes [11]. Its notable features include the ability to capture snapshots of a student's progress on an assignment as the solution is developed over time, and the implementation of an exception checker that can collect data about common failures that occur in student code. Our approach differs from Marmoset's in the inclusion of a report generation engine, user-defined reports over the entire data model (limited by role-based access control), a WYSIWYG report designer with live data preview capabilities, and support for extending the data model with user-written data collectors via Web-CAT's plug-in mechanism.

Mierle, Laven, et al. investigate mining CVS repositories used to store students' code during development [8]. The authors analyze features such as the amount of code written, the extent of changes made between each commit, and the length of time between those commits. They attempt to draw a correlation between these metrics and a student's performance on an assignment.

While the design and goals of the work done by these two groups may differ from ours, it does give some insight into the type of data that would be useful to make available through the reporting engine so that Web-CAT can support similar types of analyses.

3 Design and Implementation

The first step to designing this project was to select the reporting engine that would be integrated into Web-CAT. The primary requirements were that it be open-source and implemented in Java. The three major candidates were JasperReports [6], JFreeReport [10], and BIRT. After examining each in detail, we chose BIRT for its extensibility, customizability, its rich user-friendly report designer, and other desirable features such as JavaScript support and the ability to generate interactive charts in SVG format.

3.1 Exposing the Web-CAT Data Store

As described earlier, Web-CAT collects a large amount of data that would be useful to compile into a report. A database contains information about assignments such as their names, descriptions, due dates and scoring guidelines. Statistics computed during the evaluation of a student's submission by grading plug-ins are written to a file that holds key/value pairs for properties such as code size, code coverage, execution time, test results, run-time errors, and static analysis results. Thus, it is not adequate simply to connect the reporting engine to the underlying SQL database using one of the standard BIRT data sources, because much of the information is not stored there.

Given this disparity in data storage, it is essential that we provide a uniform method for the reporting engine to query the data from Web-CAT. Further, since Web-CAT's plug-in-based architecture allows instructors to add their own plug-ins that collect additional data, a method with built-in extensibility is critical. The user designing the report should not have to be concerned that the data of interest comes from a database or from a field in a grading properties file. We accomplish this by providing our own extensions to BIRT that define a custom data model that accesses data in a Web-CAT-friendly manner. These extensions integrate fully with the BIRT report designer so that end-users can create report templates for Web-CAT as easily as they would for any conventional data source.

Web-CAT is built on top of Apple's WebObjects framework, which simplified the implementation. Each table in Web-CAT's database is shadowed by a Java class with accessor methods corresponding to the columns in the table. These classes can also be extended to compute dynamic properties as well. Then, values can be identified using a "key-value coding" notation, which uses dot-separated key paths to evaluate properties, similar to how JavaBeans properties are denoted. For example, *submission.user.user-Name* would call the `getUser()` method on the `submission` variable, and then the `getUserName()` method is called on that result. By adopting this notation when defining reports, we simplify the way that data is captured and reap the benefits from the WebObjects model, so that properties can be accessed in the same manner regardless of whether they are a database-backed property or one that is obtained from another source.

3.2 Extending BIRT to Communicate with Web-CAT

BIRT divides its data model into a two-tiered hierarchy. At the top level are "data sources." BIRT itself includes support for a handful of data sources, ranging from XML and CSV files to JavaScript and JDBC connections. Properties associated with the data source might be the path to a data file (in the case of XML or CSV), or the server name and login credentials (for JDBC). Then, in a report template, each instance of a data source contains one or more "data sets." A data set represents a query into that data source and defines the columns that will be retrieved for each row of the result set.

None of BIRT's built-in data sources are appropriate for interfacing with the Web-CAT object model, so we have extended it by adding a Web-CAT data source and data sets. In this case, the data source is simply a representation of the instance of Web-CAT under which BIRT is running, and has no associated properties of its own. A Web-CAT data set has two principal properties. The first is the type of object in the Web-CAT object model that it expects to receive. Each object in this set will correspond to a row in the result set. Second, the data set defines a list of columns for the report. Each column has three parts: a symbolic name that will be used to refer to it elsewhere in the report, the data type of the column, and the key path to evaluate to retrieve the value for that column. These key paths are evaluated using objects of the data set's object type as their root; in other words, if *x* is an object being fed into the report and *key.path* is one of the column key paths, then the result is the evaluation of *x.key.path.*

We do not limit the user to simple key paths, however. A column expression can be written in OGNL [1], which is syntactically a proper superset of standard key paths that provides additional features such as enhanced list operations and pseudo-lambda expressions. This added flexibility can be useful when it is necessary to perform additional small calculations, such as aggregations, for which a simple key path would be insufficient. In addition, BIRT uses the Rhino [9] implementation of JavaScript so that scripts can be written in a report to transform the data further as necessary.

3.3 Maintaining a Library of Report Templates

In order for users to gain the most benefit from the Web-CAT reporting engine, they should be encouraged to share the report templates that they create so that other instructors can use the templates to generate reports for their own classes. A similar model is currently in use for Web-CAT's grading plug-ins, which can either be private to the user who created them or published for anyone's use, and are annotated with metadata that provides human-readable names, descriptions, and parameter information.

We mimic this design by providing a template library to which users can upload the report templates they create. Rather than designing a report with a fixed course, assignment, or other query in mind, the templates are constructed so that they process a set of a particular type of objects in the Web-CAT object model (such as submissions); then, users can visualize data from a course or assignment of their choosing. When a report template is chosen for generation, Web-CAT will ask the user to specify a query that will be used to determine which objects are passed to the report.

3.4 The Life Cycle of a Report

A report on Web-CAT goes through three phases: the report template, the intermediate report document, and the final rendered report. The report template (called a "report design" by BIRT), as one would expect, contains the data set definitions, layout, charts, and tables that make up the report. Report templates can be produced by BIRT's graphical report designer application. Once designed, the template can then be used by many instructors to create reports for different courses or assignments.

When an instructor requests a report using a given report template, BIRT generates an intermediate file that it refers to as the "report document." This is an internal representation of the report that contains all of the data retrieved from the data source, but that is not yet rendered in a human-readable format. Our design caches these report documents internally for later use.

In the final phase, the intermediate document is rendered into one of several formats and presented to the user. These formats include HTML, PDF, and Microsoft Excel. Support also is provided to extract the data from a report into CSV files to be analyzed using an external tool if desired.

There are two motivations for caching the intermediate report document as opposed to running the generation and rendering phases as a single task. First, the rendering phase of the report typically requires only a negligible amount of time when compared to the generation phase, which can require seconds or even minutes to complete, depending on the nature of the data being processed. If the user wishes to re-render the report in a different format, either immediately after it is generated or later, starting from the cached report document greatly speeds up this process. Second, a report generated at a particular time should represent a snapshot of the data on Web-CAT at that time. If the report were regenerated each time it was viewed, it would diminish the usefulness of using the reports for comparative analysis.

4 Data Mining Applications

Prior to the implementation of the reporting engine, there were many questions regarding student performance and habits about which we could only make educated guesses based on anecdotal evidence. With the entire data store of Web-CAT now exposed, we are in a much better position to answer these questions with quantifiable evidence. In the sections below, we describe some of these questions, examine how the reporter can be used to mine data from an assignment, and discuss whether the results either reinforce or challenge our original assumptions. All of the reports shown below were generated with data mined from the same assignment in a single course unless indicated otherwise.

4.1 Number of Submissions versus Final Score

Do students who make more submissions receive higher scores than those who make fewer? Web-CAT allows students to make multiple submissions before the deadline, so that students can get feedback early and participate in more feedback/revise/resubmit

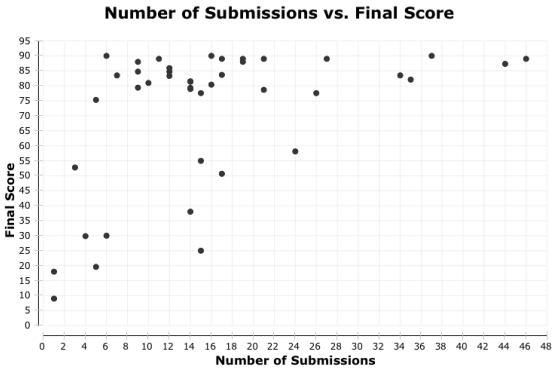


Figure 1.

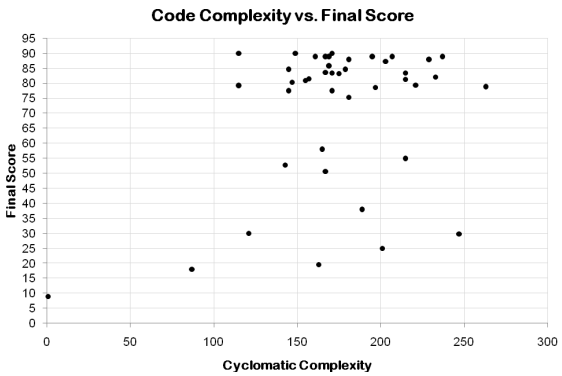


Figure 2.

cycles. Students who make few submissions could be either advanced students who quickly arrive at the correct solution, or poorer students who start late and give up quickly. Students who make an excessive number of submissions may not be thinking about the problem critically, and may just be spinning their wheels.

Figure 1 shows a scatter plot where the horizontal axis is the total number of submissions that a student made and the vertical axis is the final score that each student received on the last submission. Students who made four or fewer submissions scored poorly. Beyond that, grades improved dramatically. Those who made many more submissions were also able to achieve a high score with which they were satisfied. There is a clear partitioning of the student scores at 71, which separates students who passed the assignment (a letter grade of C or better) and those who failed. A chi-square test of number of submissions between the two groups indicates a significant difference ($df = 1$, chi-square = 7.6, $p = 0.006$, $\alpha = 0.05$), with students failing the assignment making fewer submissions (a mean of 9.5, compared with 18.3 for students who passed).

4.2 Code Complexity versus Final Score

Is there a correlation between the complexity of a student's submission and the score received? Here, we have used McCabe's cyclomatic complexity measure [7], although other measures are possible. It is even possible for instructors to devise their own measures and write a Web-CAT plug-in to collect custom data as part of the grading process, which then becomes directly accessible in custom reports.

We hypothesize that a graph of complexity vs. score would resemble a bell curve, where low-complexity submissions indicate a student failed to write an adequate solution, while excessively complex solutions indicate a student who may have written too much code without a proper understanding of the problem, perhaps using a "band-aid" approach.

Figure 2 shows a scatter plot where the horizontal axis is the cyclomatic complexity of the student's final submission and the vertical axis is the final score. Although outliers on both ends tended to score poorly, no clear bell shape emerges and low-scoring submissions are distributed throughout the range. A chi-square test fails to show any significant difference between students who achieved passing scores and those that did

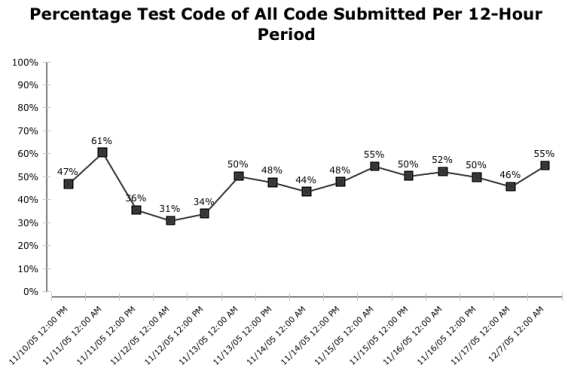


Figure 3.

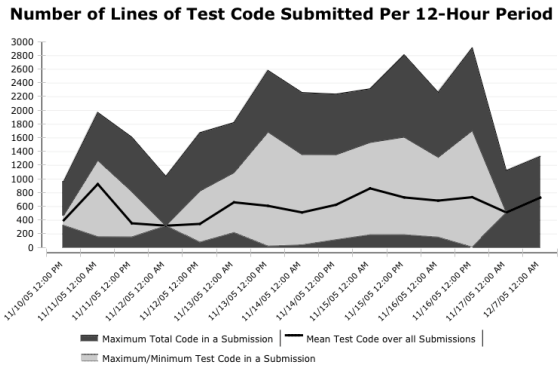


Figure 4.

not. This result suggests re-running the report over multiple assignments—possibly over multiple semesters—might provide a better visualization of any trend that is present.

4.3 Early versus Late Testing: Student Habits

When do students begin writing unit tests for the components in their solutions—incrementally as they write the code, or only after writing the majority of their code? This question interests us because we require students to test thoroughly the code that they write in our CS1 and CS2 courses. To answer this, we can examine the submissions made on each day as an assignment deadline approaches and examine the total number of lines of test code among them versus lines of non-test code.

Figures 3 and 4 show the results. Figure 3 is a line graph that shows, for all submissions occurring in each 12-hour period, the percentage of all code submitted that was unit test code. Figure 4 is a more detailed breakdown of this same information, displaying the minimum, maximum, and average number of lines of test code submitted in each 12-hour block, as well as the maximum number of lines of all code submitted for comparison. Although there is a slight dip approximately five days before the assignment is due—caused by one student, as indicated in Figure 4—these two plots give a strong indication that students overall are testing their work incrementally, rather than waiting until they have finished their solutions before beginning to write their own tests.

We can also examine assignments at the beginning and end of a course offering to see if testing habits improved as time went on. The chart in Figure 3 reflects the final assignment in a course. Figure 5 shows a similar chart for the first assignment given in the same course. The plot for the first assignment indicates that students were writing tests incrementally at the start of the course, as the instructors intended.

4.4 Early versus Late Testing: Affects on Final Score

Do students who begin testing their code early perform better on the assignment overall than those who put off testing until much later? This question arises when instructors consider encouraging students to use test-driven development (TDD), a strategy where they incrementally write tests along with their solution, writing each test just before completing the corresponding part of their implementation. Anecdotal evidence might

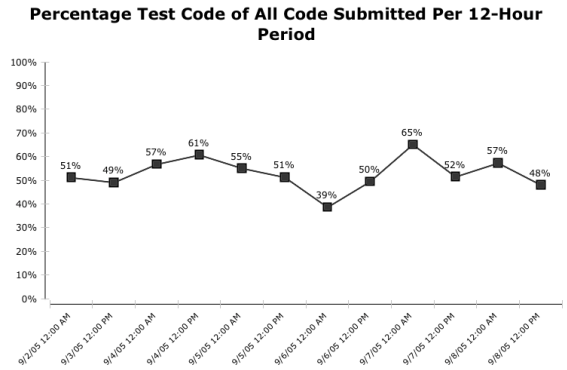


Figure 5.

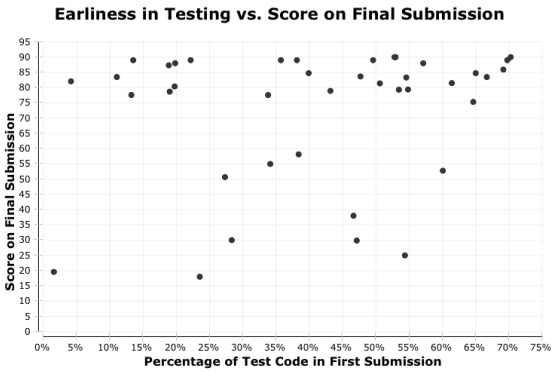


Figure 6.

suggest that students score higher if they test early, and experimental evidence indicates that if they are graded on how well they test, they produce significantly fewer bugs and score higher [3].

To answer this question, we first need to be able to quantify how early a student began writing tests. For simplicity, we will define this as the percentage of test code in their first submission. We rationalize this choice by stating that students should be doing their own testing as they write their solution piece by piece, even before having a sufficiently complete attempt to submit for grading. Students who do not start writing tests early will have a low percentage, while students who do should have a higher one.

A number of other more detailed metrics could be explored instead. Rather than examining the percentage of test code, one could combine this in some fashion with the percentage of test code that was actually executed according to code coverage tools, the degree of code coverage achieved by executing the tests, or a time-based approach determining when, if at all, a student passes a pre-defined threshold of testing.

Figure 6 shows a scatter plot where the horizontal axis is the percentage of test code in each student's first submission and the vertical axis is the score received on his or her final submission. Because the earliness of testing is based on the percentage of test code found, *higher* numbers toward the right indicate *earlier* testing.

A chi-square test between students who achieved passing scores and those that did not fails to show any significant difference in the percentage of test code in the first submission. However, the plot shows an interesting gap in the percentage of test code around 25-30%. Indeed, using the likelihood-ratio chi-square to produce an optimum partitioning by percentage of test code leads to a split point of 25%. We can consider students below this threshold as those who "test less" early, and those above the threshold as those who "test more" early. One possible explanation is that students with greater programming experience—who also are more resistant to test-first development practices, and end up in the "test less" group—manage to do well on assignments at this level. Another is that students who did little testing early quickly learned that it was to their advantage to do more and improved on this before their final submission. We could delve further into either theory by generating another report with the progression of

percentage of test code from the first submission to final submission, breaking it down by students who started with little testing vs. students who started with more.

5 Future Work

We realize that many instructors would prefer to make use of an existing set of report templates rather than undertake the task of creating their own. While we cannot predict the exact needs of everyone who uses Web-CAT, it behooves us as the system designers to develop as comprehensive a library of templates as possible so that other instructors can immediately benefit from the system. As is the case with Web-CAT's grading plug-ins, we can make use of Web-CAT's auto-updating feature to distribute new or revised report templates to users as we would distribute system component updates.

At the same time, many of the reports discussed in Section 4 required significant JavaScript code and OGNL expressions to be written into the data set descriptions. BIRT provides these tools for creating reports that are more intelligent, and they were used here to transform the data in a way that was appropriate for the view that we desired. However, using these features necessarily increases the learning curve for new users. We can simplify this either by extending the Web-CAT data model as deemed necessary, or by providing utility code to assist with common operations and transformations, either in the form of JavaScript "snippets" that can be dropped into a report or server-side Java code that can be called from within a report.

By doing this, common data mining techniques such as regression, classification with decision-trees, and others could be embedded so that a user could integrate their results into a report more easily. Even without these more advanced features, however, it is easy under the current system to export various cuts and views of the Web-CAT data store into CSV files that could be analyzed by more specific data mining tools off-line.

Another topic that the reporting tool opens up is the possibility of doing *predictive analysis* based on the data mined from Web-CAT. Instructors could use data about past assignments to predict future performance, going as far as breaking down an assignment into specific problem areas and identifying students who had particular trouble with certain concepts. These ideas would benefit further from the ability to set up a report to be automatically generated on a recurring schedule. Trends in data could be predicted in one run of the report and then compared to the actual outcome of the next run. E-mail notifications to instructors or even to students could be used to send alerts of at-risk situations, while students are still developing their solutions before an assignment is due.

6 Conclusions

This paper has presented a solution to the issue of accessing and analyzing the large amount of data that is generated from an automated grading system. We have discussed the design and integration of a report generator with an existing automated testing system, Web-CAT, and shown how complex reports can be used to mine data from student coursework to answer deep questions about their performance and habits.

As of this writing, the implementation of the reporting engine was only recently completed. As such, no formal evaluation of the system has yet been undertaken, but the potential for this type of system should be apparent. Instructors and computer science researchers now have the ability to both generate simple grade reports and distributions, as well as perform empirical analysis on their coursework to answer questions as complicated as those described above. The answers to these types of questions may aid an instructor in improving their curriculum.

Due to the flexibility of both Web-CAT and BIRT, there are also potential opportunities outside the realm of computer science education. Web-CAT is in essence a large plug-in manager and integrating BIRT extends that idea to include report templates. Plug-ins can be devised that would allow for any type of analysis on code, and the results of that process could then be rendered for study. An example might be an in-depth study of how a department in a corporation complies with the company's coding standards. An evaluation tool of this sort would be beneficial in any case where statistical data is desired out of a large collection of source code. Further, applying these ideas to general-purpose course management systems could lead to similar capabilities for other disciplines.

References

[1] Blanshard, L. and Davidson, D. *OGNL: Object-Graph Navigation Language,* 2008. Available at: <http://www.ognl.org>

[2] The Eclipse Foundation. *BIRT Project: Business Intelligence and Reporting Tools,* 2008. Available at: <http://www.eclipse.org/birt/ >

[3] Edwards, S. H. Improving student performance by evaluating how well students test their own programs, *Journal of Educational Resources in Computing,* 3(3): 1-24 (2003).

[4] Edwards, S.H. *Web-CAT Wiki*, 2008. Available at: <http://web-cat.org>

[5] Edwards, S. H., M. Pérez-Quiñones, M. Phillips and J. RajKumar. Graphing Performance on Programming Assignments to Improve Student Understanding, *Proceedings of the International Conference on Engineering Education,* 2006.

[6] JasperSoft Corporation. *JasperReports,* 2008. Available at: <http://jasperreports.sourceforge.net/>

[7] McCabe, T. A Complexity Measure*, IEEE Transactions on Software Engineering,* vol. SE-2, no. 4, 1976, pp. 308-320.

[8] Mierle, K., K. Laven, S. Roweis and G. Wilson. Mining Student CVS Repositories for Performance Indicators, *2005 International Workshop on Mining Software Repositories,* ACM, 2005, pp. 1-5.

[9] Mozilla.org. *Rhino: JavaScript for Java*, 2008. Available at: <http://www.mozilla.org/rhino/>

[10] Object Refinery Limited. *JFreeReport,* 2008. Available at: <http://www.jfree.org/jfreereport/index.php>

[11] Spacco, J., J. Strecker, D. Hovemeyer and W. Pugh. Software Repository Mining with Marmoset: An Automated Programming Project Snapshot and Testing System, *2005 International Workshop on Mining Software Repositories,* ACM, 2005, pp. 1-5.

Analyzing Rule Evaluation Measures with Educational Datasets: A Framework to Help the Teacher

Sebastian Ventura, Cristobal Romero and Cesar Hervás
{sventura, cromero, chervas}@uco.es
Department of Computer Science and Numerical Analysis. University of Cordoba.
Campus de Rabanales, edificio "Albert Einstein", 14071, Córdoba, España.

Abstract. Rule evaluation measures play an important role in educational data mining. A lot of measures have been proposed in different fields that try to evaluate features of the rules obtained by different types of mining algorithms for association and classification tasks. This paper describes a framework for helping non-expert users such as instructors analyze rule evaluation measures and define new ones. We have carried out several experiments in order to test our framework using datasets from several Cordoba University Moodle courses.

1 Introduction

Rule discovery is one of the most popular data mining techniques, especially in EDM (Educational Data Mining), because it shows the teacher information that has been discovered and does so in an intuitive way [7]. The rules discovered by a data mining method must be of interest for end-users in order to be considered useful. Today, measuring the interest in the rules discovered is an active and important area of data mining research [2]. Its main objective is to reduce the number of mined rules by evaluating and post-pruning the rules obtained in order to locate only the most interesting rules for a specific problem. This is very important for a non-expert user in data mining, like a teacher. In order to evaluate rules there are a wide range of measures [8]. Rule evaluation measures are used for ranking and selecting rules according to their potential interest for the user [8]. However, as there is no single measure that is optimal in all application domains and some authors propose new measures, it is necessary to evaluate the performance of each measure for each different domain and/or dataset [2].

In the educational domain, the study of rule evaluation measures is a very important issue, although there are only a few papers about it. Ma et al. [4] select weak student ranking association rules by a scoring function that uses several measures. Minaei-Bidgoli et al. [6] discover interesting contrast rules using different evaluation measures. Romero et al. [7] select a group of rule evaluation measures to select the best prediction rules. Merceron and Yacef [5] revisit the measuring of degrees of interest in their strong symmetric association rules in educational data.

In this paper, we propose a framework for analyzing rule evaluation measures. The paper is organized as follows: section 2 surveys rule evaluation measures, section 3 describes the rule evaluation framework, and section 4 describes experimental tests using datasets from Moodle courses' student usage information. Finally, in section 5 the conclusions and further research are outlined.

2 Rule Evaluation Framework

Most of the general-purpose data mining tools such as Weka [9] discover (association, prediction, classification) rules and show them to the user together with traditional evaluation rule measures such as accuracy, support and confidence. But these tools do not show the values of other important rule evaluation measures [8], nor do they permit their comparison. Thus it is very hard for a user that is not an expert in data mining (e.g. a teacher) to appropriately use the great variety of existing rule evaluation measures to select only the most interesting rules. In order to try to resolve this problem, we have developed a rule evaluation framework (see Figure 1).

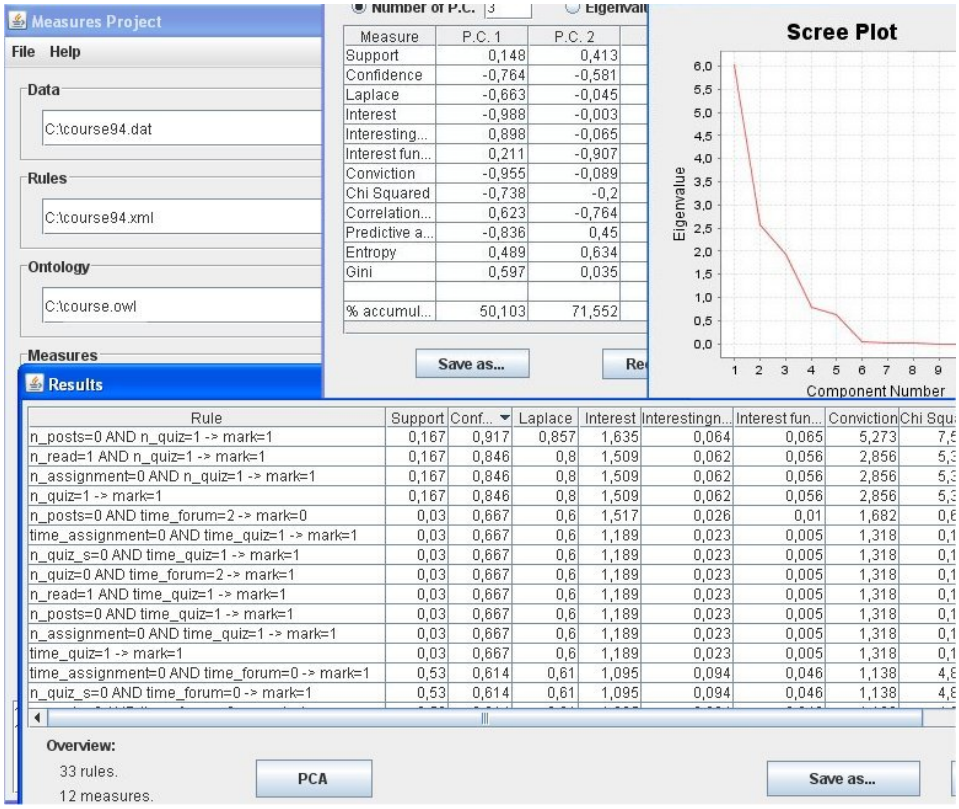


Fig. 1. Main window of the application and windows with the rules and values of the measures.

In order to use our framework, the user/teacher has to select 2 obligatory files (data and rules) and can also select 2 optional files (measures and ontology). The data file is a local file or URL (Uniform Resource Locator) that contains the dataset. This file can have CSV (Comma-Separated Values) format or Weka format [9]. The rules file is also a local file or URL that contains the rules discovered by the mining algorithm. It has to have PMML (Predictive Modeling Markup Language) format that is an XML-based language which provides a way for applications to define statistical and data mining models and to share models between compliant applications. The measure description file is an optional local file with XML format that contains the definition of the measures in Latex equation format. Thus the measures are defined using not only mathematics latex symbols and functions, but also probabilistic and contingency table symbols, such as n(A), n(C), N,

P(A), P(C/A), etc. as well as all the availables measures for defining new measures. The framework provides a default measure description file with over 40 measures already defined so that the teacher can use them directly with no need to define them. We have also developed a wizard and an equation editor using Latex equation format in order not to have to write out this XML file by hand, since this could be a difficult task for a teacher. Using this wizard, the teacher can define brand-new measures easily by only following several guided steps and the equation editor. Finally, the ontology file is an optional local file. It uses OWL (Ontology Web Language) format to define the specific domain of the data and rules used. Then, after the user/teacher has chosen the previous input files; the application calculates the values of each measure for each rule coming from the data file. Next, all the rules and all the measures are displayed for the teacher in the *results* window (see Figure 1 down). For each rule, the elements of rule antecedents and consequents are displayed as well as the calculated values of each evaluation measure. The rules can be sorted by any of the measures by simply clicking on the header of a specific column so that the teacher can compare different ranks depending on the measure used. Furthermore, if the user has selected an ontology file, then the OWL file will be visualized graphically so the teacher can interpret/understand better the meaning of the rules in that domain.

Finally, we have also developed a PCA (Principal Component Analysis) module [1] in order to help the user/teacher to group and reduce the number of measures used. Each principal component is unrelated and corresponds to orthogonal directions in the newly generated search space. In our framework, the teacher can execute PCA starting from the results windows by pressing the *PCA* button (see Figure 1 down). A new window appears (see Figure 1 up) in which the user/teacher can select the number of principal components or the maximum eigen value, and has the option of showing the scree plot and the coefficients of the measures in each PC. Then the communality values of each principal component for each measure are shown along with the scree plot. Using the scree plot and the eigenvalues, the user/teacher can select a number of principal components, normally those with eigenvalues greater than 1 or when the inclination of plot starts to decrease. Then, the teacher can group each measure into one principal component using the communality values for each measure. In order to do so, the teacher has to assign or classify each measure in the component where it shows the highest absolute value.

3 Experimental results with Moodle courses

We have carried out several experiments in order to test our framework using educational datasets. We have used 4 dataset files obtained from 4 Moodle courses with about 80 students in each course. The courses are computer science technical-engineering second-year courses in Cordoba University. We have preprocessed these students' usage data that is stored in a Moodle database. First, we have filtered only the information about the 4 courses activities that interest us, such as assignments, forums and quizzes. Next, we have created a summarization table that integrates this information at student level (number of assignments done, number of messages sent/read to/in the forum, number of quizzes taken/passed/failed, total time used on assignment/quiz/forum, and student's final mark). We have discretized all the numerical values in order to increase interpretation

and comprehensibility since categorical values (low, medium and high) are more familiar to teacher s than precise magnitudes and ranges. Finally, we have saved them in 4 data files, one for each course. Before using our framework, we applied the Apriori-C algorithm [3] to discover Class Association Rules (CARs) from previous Moodle data. Apriori-C is a well-known algorithm for discovering association rules for classification. In our experiment, the class is the mark attribute. In this way, the teacher can obtain rules that show relationships between Moodle activities that influence the mark obtained by the students [15]. Specifically, we have executed Apriori-C algorithm over the 4 course dataset files with a minimum support of 0.03 and a minimum confidence of 0.6 as parameters. Next, the class association rules obtained are saved into a PMML file. In the first column of Table 1, we can see the number of rules obtained for each dataset.

Table 1. Comparison of %Variance with the principal components using 12 measures.

Dataset	Number of rules	1 P.C.	2 P.C.	3 P.C.	4 P.C.
Course 1	33	50,103	71,552	87,693	94,235
Course 2	36	36,001	65.411	78,382	88,414
Course 3	32	34,383	60,486	78,961	87,443
Course 4	29	36,547	60,518	80,733	89,098

Next, we have used our framework to obtain values for all evaluation measures beginning with the Moodle datasets and PMML rules files. In this experiment, we have used only 12 measures: chi-squared, correlation coefficient, predictive association, entropy, support, confidence, laplace, interest, interestingness, gini, interest function and conviction. We have chosen these measures specifically because they are some of the most representatives [8]. So we have obtained the values of the 12 measures for all the rules and then we have applied PCA with 1, 2, 3 and 4 principal components in order to see how many components or groups all these measures can be grouped into. In Table 1, we can see the amount of variance obtained for each principal component from the rules discovered in each dataset. The results obtained show that we can select 3 principal components since they store between 80%-90% of the variance of the data. So, we can use these components as new evaluation measures in which almost all the information provided by the original measures is included. In this way we can reduce the number of measures used from the original 12 measures to 3 new meta-measures. The teacher could define three new measures using the editor, one for each PC using the coefficients of the measures in each PC**.** For example, the 2nd PC in course 1 could be defined as the following new measure:

```
$PC2=0.565*Support-0.362*InterestFunction$
```

4 Conclusions and Future Work

In this paper we have described a specific framework for analyzing rule evaluation measures. We have shown how a user/teacher can use it together with Moodle course datasets in order to: obtain the values of the measures of the rule discovered by a rule mining algorithm, sort and select the rules according to the values of any specific

measure, compare and group the measures using PCA and define new measures using a wizard and an equation editor. Currently we are working on other techniques for comparing rule evaluation measures. For example, correlation techniques in order to see what measures are correlated. In the future, we want to work with subjective and semantics-based measures. Our objective will be to add subjective restrictions to our framework user that take into account information about the domain. For example, restrictions to different granularity levels to only show rules about relationships between activities, or relations between chapters or between courses. Finally, we also want to develop brand-new semantic-based measures that can use the domain information of the OWL files. In this way, we could create new measures specifically geared toward each application domain or dataset.

Acknowledgments. The authors gratefully acknowledge the financial subsidy provided by the Spanish Department of Research under TIN2005-08386-C05-02 projects. FEDER also provided additional funding.

References

[1] Fukugana, K.: *Introduction to Statistical Pattern Recognition*. Elsevier. 1990.

[2] Geng, L., Hamilton, H.J.: Interestingness Measures for Data Mining: A Survey. ACM Computing Surveys, Vol. 38, No. 3 (2006) 1-32.

[3] Jovanoski, V., Lavrac, N.: Classification Rule Learning with APRIORI-C. In Proc. Conf. Artificial Intelligence (2001) 44-51.

[4] Ma, Y., Liu, B., Kian, C.K. Wong, Yu, P.S., Lee, S.M.: Targeting the right students using data mining. In Proc. Conf. Knowledge discovery and data mining, Boston (2000) 457-464.

[5] Merceron, A., Yacef, K.: Revisiting interestingness of strong symmetric association rules in educational data. In Proc. of Int. Workshop on Applying Data Mining in e-Learning, Creete, Greece (2007) 3-12.

[6] Minaei-Bidgoli, B., Tan, P-N., Punch, W.F.: Mining Interesting Contrast Rules for a Web-based Educational System. In Proc. Int. Conf. on Machine Learning Applications. Louisville, USA (2004) 320- 327.

[7] Romero, C., Ventura, S. de Bra, P.: Knowledge Discovery with Genetic Programming for Providing Feedback to Courseware Author. UMUAI. Vol. 14. No. 5 (2004) 425-465.

[8] Tan, P., Kumar, V.: Interesting Measures for Association Patterns. In Proc. KDD Workshop on Postprocessing in Machine Learning and Data Mining, Boston, USA (2000) 1-9.

[9] Witten, I.H., Frank, E.: Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann (1999).

Mining and Visualizing Visited Trails in Web-Based Educational Systems

Cristóbal Romero [1], Sergio Gutiérrez[2], Manuel Freire[3] and Sebastián Ventura[1]
cromero@uco.es, sergut@lkl.ac.uk, manuel.freire@uam.es, sventura@uco.es
[1]Department of Computer Science, Córdoba University, Córdoba, Spain
[2] London Knowledge Lab - Birkbeck College, London, UK
[3] EPS-Universidad Autonoma de Madrid, Madrid ES-28049, Spain

Abstract. A data mining and visualization tool for the discovery of student trails in web-based educational systems is presented and described. The tool uses graphs to visualize results, allowing non-expert users, such as course instructors, to interpret its output. Several experiments have been conducted, using real data collected from a web-based intelligent tutoring system. The results of the data mining algorithm can be adjusted by tuning its parameters or filtering to concentrate on specific trails, or to focus only on the most significant paths.

1 Introduction

The concept of trails is closely related to the concept of curriculum. A curriculum is essentially a course or learning plan. A trail is the plan or route that a learner follows within a curriculum element, such as a subject, a competence or a learning goal [11]. Data mining techniques have been used to discover the patterns of sequences or visited trails of students from log files [10]. Server log files store a vast amount of information that contains the page requests of each individual user/student. This information can be seen as a per-user ordered set of web page requests from which it is possible to infer user navigation sessions. Lessons learnt from examining log files can then be fed back into the learning environment for optimization purposes. The extraction of sequential patterns or trails has been proven to be particularly useful, and has been applied to many different educational tasks. For example, trails have been used to evaluate a learner's progress, and to adapt and customize resource delivery [13]. Other uses include comparing trails to expected behavioral patterns (specified by the instructor, designer or educator) that describe an "ideal" learning path [8], providing indications on how to best to organize an educational web space, suggesting alternative paths to learners who share similar characteristics [4], the generation of personalized activities for different groups of learners [12], supporting the evaluation and validation of learning site designs [7], the identification of interaction sequences which suggest problems and/or identify patterns that are markers of success [5], supporting navigational learning, or testing the effectiveness of certain trails of learning objects [11].

In this work, we describe a tool for mining and visualizing visit trails in web-based educational systems. The next section introduces the functionality of the tool to mine and visualize trails from user logs. Experimental results achieved with real data obtained from a web-based intelligent tutoring system are then discussed. Finally, conclusions and future work are presented.

2 A tool for Mining and Visualizing Educational Trails

We have developed a mining and visualizing tool in order to help instructors to discover the most visited trails. Course authors or instructors can execute the tool whenever enough usage information from students has been collected. First, the user has to create a data file from the web-based educational system's log files. The tool's user has to indicate the name, type of database, user, password, server and port. Optional restrictions such as a time period (a start date and an end date, empty by default) and session time in minutes (25 minutes by default) can also be indicated. Then, the system creates a .dat file using the Weka format. This file contains all the visited links and the total number of times that students use each of them.

Next, the user can apply a web data mining algorithm to locate navigation sessions and trails. Currently, we have implemented the HPG (Hypertext Probabilistic Grammar) model to efficiently mine the trails [1]. HPG uses a one-to-one mapping between the sets of non-terminal and terminal symbols. Each non-terminal symbol corresponds to a link between web pages. Moreover, there are two additional artificial states, S and F, which represent the start and finish states of the navigation sessions. The number of times a page was requested, and the number of times it was the first and the last page (state) in a session, can easily be obtained from the collection of student navigation sessions. The number of times a sequence of two pages appears in the sessions gives the number of times the corresponding link was traversed. The aim is to identify the subset of these trails that correspond to the rules that best characterize a student's behavior when visiting the web-based course. A trail is included only if its derivation probability is above the parameter λ (cut-point). The cut-point is composed of two distinct thresholds (support and confidentiality). The support value is for pruning out the strings whose first derivation step has low probability, corresponding to a subset of the hypertext system rarely visited. The confidence value is used to prune out strings whose derivation contains transitive productions with small probabilities. Therefore, in order to use the HPG algorithm, the user has to select a *.dat* file, set three parameters (α, support and confidence) and the desired name of the routes file that will be created. This output text file will contain all the routes and the probabilities of the sequences to go from one node to another node. The user can also reduce the size of the generated route files by using three types of filters. The *accumulated transition probability* filter eliminates sequences that have a value lower than the user-specified value (between 0 and 100%). The *non-accumulated transition probability* filter is similar to the previous one, but does not accumulate the probability values. The *route length filter* limits the length of the routes to a maximum value, specified by the user (integer greater than 1).

Then, we visualize the resulting trails as a graph (considering each hypertext page as a node and each hyperlink as an edge) in order to display the amount of traffic for each route to instructors (see Figure 1). This graph is zoomable, and the nodes can be manually displaced, allowing users to modify the automatically generated layout or examine the graph at varying degrees of detail. The application also shows, in different panes, the name of the nodes (reduced and full name of the nodes), the routes in text mode, and the used data file.

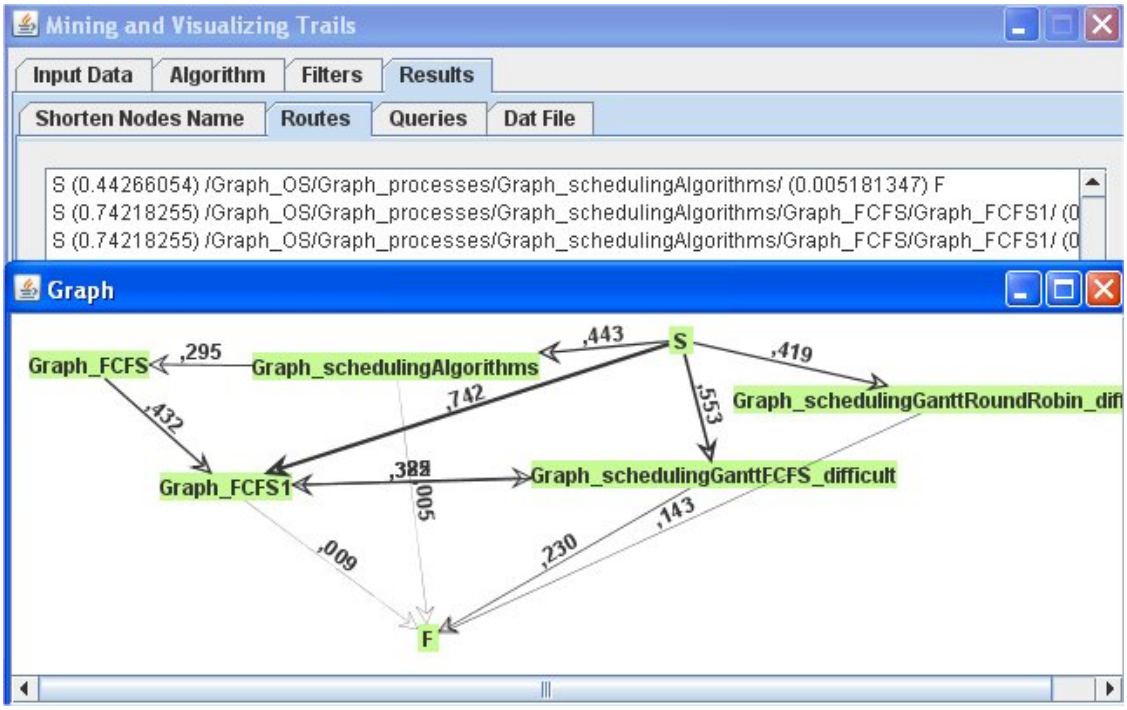


Figure 1. Results showing routes in text mode and in the graph visualization window.

In Figure 1, each node represents a web page, and the directed edges (arrows) indicate how the students have moved between them. The tool's graph visualizations are generated using the CLOVER framework, described in greater detail in [3]. Edge thickness varies according to edge weight; this allows users to quickly focus on the most important edges, ignoring those that have very low weights. In addition to line widths, numerical weights are also available. This information can be useful to a learning designer in different ways. First, it can be used as a limited *auditing tool*, providing a deeper understanding of the learning paths effectively followed by the students. Additionally, comparing this information with expected *a priori* paths allows the designer to refine the sequencing strategy. The results in the graph can show information that was not know in the first place, e.g. which activities are the most difficult, which are easier than expected (shown as more common transitions), etc.

3 Experimental Results

This study uses real data collected from a web-based intelligent tutoring system [9] for the domain of Operating Systems. Although the original log file contains sessions from 88 students that used the system in 2006, the study covers only the subset of "good users" (those with more than two sessions). The study covers data from 67 students, with 754 sessions (using 25 minute timeouts) and 1121 records in total. We have carried out several experiments focused on HPG's sensibility to its parameter values in order to obtain different configurations of the graph (number of nodes, links, routes, and average route length). Results with varying parameters are displayed in Table 1.

Table 1. HPG: Comparison varying alpha and cut-point parameters.

Alpha	Support	Confidence	Nodes	Links	Routes	Avg. Route Length
0	**0.08**	**0.3**	6	8	4	4.5
0	**0.08**	**0.2**	48	123	97	16.26
0	**0.08**	**0.18**	117	340	76443	32.19
0.5	**0.5**	**0.7**	4	4	2	3.0
0.5	**0.3**	**0.7**	19	32	17	2.88
0.5	**0.1**	**0.7**	136	244	144	2.83
0.5	**0.5**	**0.2**	4	6	6	3.66
0.5	**0.3**	**0.2**	62	177	2190	14.45
0.5	**0.1**	**0.2**	162	551	46913	18.62
1	**0.7**	**0.7**	11	17	9	2.88
1	**0.5**	**0.7**	30	54	28	2.92
1	**0.3**	**0.7**	89	167	94	2.92

Support and confidence thresholds give the user control over the quantity and quality of the obtained trails, while α modifies the weight of the first node in a user navigation session. In Table 1, the support must be set very low in order to obtain routes with $\alpha = 0$. This is due to the fact that there are few start nodes. It shows that students have started their sessions in different nodes, and none of these have a significantly higher probability. This changes as α increases, since there will progressively be more visited nodes. The number of routes, nodes and links is increased as time support is decreased. On the other hand, the number of resulting nodes, links, routes and average route lengths is greatly increased when the confidence value is decreased. This effect is more evident on links and routes. This can be traced to the fact that the confidence threshold prunes the intermediate transactions that do not have a derivation probability above the cut-point. It must be noted that the user of the HPG algorithm can use both the alpha, support and confidence thresholds, and the three available filters, in order to obtain a suitable number of trails. The learning designer must work with the course lecturer in order to tune these parameters to a particular community of learners. Then, combining the information on the table with that displayed in the graph, the instructor can focus on the most visited routes in order to make decisions on the organization of the educational web space, or recommend paths and shortcuts to learners.

4 Conclusions

This paper has described a data mining and information visualization tool that aids authors and instructors to discover the trails followed by students within web-based educational systems. The resulting networks are then visualized using a graph representation with edges of varying thickness, which is more compelling to non-specialized users than textual output. Future plans include the addition of other sequential pattern mining algorithms such as AprioriAll and PrefixSpan. Our goal is to use the tool to provide personalized trails to students, delivering on the promise of personalized learning within adaptive e-learning systems.

Acknowledgments. The authors gratefully acknowledge the financial subsidy provided by the Spanish Department of Research under TIN2005-08386-C05-02 and TIN2007-64718, and the British Teaching and Learning Research under grant RES-139-25-0381.

References

[1] Borges, J., Levene, M. Data Mining of user navigation patterns. Proc. of Workshop Web Usage Analysis and User Profiling. San Diego, 2000. pp. 31-36.

[2] Dichev, C., Dicheva, D., Aroyo, L. Using Topic Maps for Web-based Education *Advanced Technology for Learning*, 2004, 1, pp. 1-7.

[3] Freire, M. An Approach to the Visualization of Adaptive Hypermedia Structures and other Small-World Networks based on Hierarchically Clustered Graphs. PhD Thesis presented at Universidad Autónoma de Madrid, 2007.

[4] Ha, S., Bae, S., Park, S. Web Mining for Distance Education. IEEE Conf. on Management of Innovation and Technology, Singapore, 2000. pp. 715–719.

[5] Kay, J., Maisonneuve, N., Yacef, K., Zaiane, O.R. Mining Patterns of Events in Students' Teamwork Data. Educational Data Mining Workshop, Taiwan, 2006. pp. 1-8.

[6] Keenoy, K., Levene, M. A Taxonomy of trails of learning objects. Trails of Digital and non-digital Los. EU Sixth Framework programme priority 2. 2004. pp. 97-106.

[7] Machado, L., Becker, K. Distance Education: A Web Usage Mining Case Study for the Evaluation of Learning Sites. In Proc. Int. Conf. on Advanced Learning Technologies, Athens, Greece, 2003. pp. 360-361.

[8] Pahl, C., Donnellan, C. Data mining technology for the evaluation of web-based teaching and learning systems. In Proc. E-learning. Montreal, Canada, 2003. pp. 1-7.

[9] Prieto Linillos, P., Gutiérrez, S., Pardo, A., Delgado Kloos, C. Sequencing Parametric Exercises for an Operating System Course. Proc. of AIAI 2006, pp. 450-458.

[10] Romero, C., Ventura, S. Educational Data Mining: a Survey from 1995 to 2005. Expert Systems with Applications, 2007, 33(1), pp. 135-146.

[11] Schoonenboom, J., Levene, M., Heller, J., Keenoy, K, Turcsanyi-Szabo, M. Trails in Education. Technologies that Support Navigational Learning. Sense Publishers. 2007.

[12] Wang, W., Weng, J., Su, J., Tseng, S. Learning portfolio analysis and mining in SCORM compliant environment. Proc. ASEE/IEEE Frontiers in Education Conference, Savannah, Georgia, 2004. pp. 17–24.

[13] Zaïane, O., Luo, J. Web usage mining for a better web-based learning environment. In Proc. Conf. Advanced Technology For Education, Banff, Alberta, 2001. pp. 60–64.

Mining the Student Assessment Data: Lessons Drawn from a Small Scale Case Study

Mykola Pechenizkiy[1], Toon Calders[1], Ekaterina Vasilyeva[1,2], and Paul De Bra[1]
{t.calders, m.pechenizkiy, e.vasilyeva}@tue.nl, debra@win.tue.nl
[1] Department of Computer Science, Eindhoven University of Technology, the Netherlands
[2] Department of Computer Science and Information Systems, University of Jyväskylä, Finland

Abstract. In this paper we describe an educational data mining (EDM) case study based on the data collected during the online assessment of students who were able to immediately receive tailored and elaborated feedback (EF) after answering each of the questions in the test. Our main interest as domain experts (i.e. educators) is in studying (by employing any kind of analysis) how well the questions in the test and the corresponding EF were designed or tailored towards the individual needs of the students. The case study itself is aimed at showing that even with a modest size dataset and well-defined problems it is still rather hard to obtain meaningful and truly insightful results with a set of traditional data mining (DM) approaches and techniques including clustering, classification and association analysis.

1 Introduction

In this paper we present a fresh small-scale EDM case study aimed at demonstrating the potential and the limitations of the traditional DM approaches and techniques [5], focusing on the problems of data redundancy and inherited correlation between many attributes that complicate the discovery of truly interesting patterns in the data.

The data in our case study comes from a real online exam (partial exam for the Human-Computer Interaction course) taken by 73 students. This exam has been organized in the form of a multiple-choice test aimed at demonstrating to the students (and teachers) what they have learnt or what (common) misconceptions they might have, and to provide yet another possibility and means for student to fill the gaps in their knowledge (or "patch" the possible misconceptions). Hence, elaborated feedback (EF) has been designed for each of the questions in the multiple-choice test.

The design of appropriate feedback is a critical issue in the development of online assessments within Web-Based Learning Systems (WBLSs). In our recent works we demonstrated the possibilities (and benefits) of tailoring the feedback that is presented to a student as a result of his/her response to questions of an online test, taking into account the individual learning styles (LSs), certitude in a response and correctness of this response [3,4]. Here, our goal is to report upon our experiences of applying different DM techniques for assessing how well the questions in the test and the corresponding EF were designed and tailored towards the students.

2 Online Assessment and Feedback Adaptation

Online assessment becomes an important component of modern education. Nowadays it is used not only in e-learning, but also within blended learning, as part of the learning process. Online assessment is utilized both for self-evaluation and for "real" exams as it tends to complement or even replace traditional methods of evaluation of the student's performance.

Feedback is usually a significant part of the assessment as students need to be informed about the results of their (current and/or overall) performance. The great variety of feedback functions and types that current system can actually support make the authoring and design of the feedback in e-learning rather complicated. In spite of the diverse interest in educational research dealing with feedback, the methods and guidelines for designing and implementing feedback in educational practice remain scarce so far [2]. This especially applies to the design of feedback in WBLSs. An important issue is that different types of feedback can have a different effect (positive or negative) on the learning and interaction processes [1].

Our studies demonstrated that knowledge of the response certitude (specifying the student's certainty of the correctness of the answer) together with response correctness helps in understanding the learning behavior and allows for determining what kind of feedback is more preferable and more effective for the students. EF may significantly improve the performance of students within the online tests [3]. We demonstrated also the potential of tailoring feedback towards individual learning styles [4].

3 Data collection and preparation

We have studied different aspects of feedback tailoring during a series of experiments in the form of eight online multiple-choice tests in the *Moodle* learning system organized as an integral part of courses (with traditional in-class lectures and instructions) at the Eindhoven University of Technology, the Netherlands during the academic year 2007-2008.

In this data-mining study we focused on the most recent online assessment (partial exam) of 73 students in the Human-Computer Interaction (HCI) course that was organized in March 2008. In some of the earlier assessments we also used feedback adaptation strategies based on the student's response correctness and certitude, and learning style.

The online test consisted of 15 multiple-choice questions. The questions were aimed at assessing the knowledge of the concepts and the development of the necessary skills (e.g., understanding the basic usability rules and problems such as consistency, mapping (between interface and real world), response time problem, etc.). For each answer students had to provide their certitude (affecting the grade) and had a possibility to request and examine the EF that could potentially help to answer related questions better.

For every student and for each question in the test we collected all the possible information, including correctness, certitude, grade (determined by correctness and

certitude), time spent for answering the question, whether feedback was requested on not, and which feedback was shown directly (if any), was recommended with which strength, and finally which one(s) were actually examined (including time spent for examining each type of feedback in seconds). Before passing the actual tests the students were asked (a few days before) to answer to the learning style questionnaire (that was not compulsory); 90% of students filled the questionnaire. The collected data was transformed into a transactional multi-relational presentation with different views for the corresponding DM tasks.

Further details regarding the organization of the test (including an illustrative example of the questions and the EF) and the data preprocessing are made available in an appendix we placed online at http://www.win.tue.nl/~mpechen/edm08/.

4 Mining for interesting patterns

Certainly, we were eager to find interesting and/or unexpected patterns in student feedback preferences or performance, in order to quantify whether feedback was helpful in answering related questions in the test, to determine if the performance and preference patterns of students with different learning styles differ significantly from each other, etc. However, first we would like to see patterns that reflect adaptation rules that effected feedback tailoring and other background knowledge as an evidence that the data confirms the system was performing according to planned behavior. We applied several DM techniques for association analysis, classification, and clustering (with and without dimensionality reduction). We describe the outcomes in the following subsections.

4.1 Classification

Instead of trying to achieve the highest possible accuracy, our goal was finding descriptive or discriminative patterns providing us with an insight and evidence supporting a particular hypothesis. One simple and straightforward approach is building a decision-tree (like C4.5) or a rule-based model (like JRip) and to analyze it.

Defining a concrete classification task we were able to investigate various patterns extracted from the classification models built on different subsets of the attributes that were selected either manually or with feature subset selection techniques. On the one hand this simple approach helps in searching evidence of the potential of hypotheses. We could see for instance from the C4.5 model that reading EF for a particular question raises the chances of answering a related question correctly and thus increases the chance of passing the whole test. On the other hand, the C4.5 classifier performs rather poorly on this dataset thus forcing us to be suspicious about any conclusion drawn from the model.

4.2 Clustering

During the test design, it is always a challenge to come up with questions that would cover the course material reasonably well, that are reasonably independent from each other (so that e.g. answering one question incorrectly would not strongly correlate with answering a large portion of the other questions incorrectly as well), that are able to

capture possible misconceptions and that satisfy a number of other desired properties. Similarly, there is a challenge in designing effective feedback and tailoring it towards an individual student and his/her current needs.

Hierarchical clustering of questions according to the student response correctness or grade produces rather intuitive dendrograms. Similarly, intuitive outcome can be achieved by clustering based on the conditional probabilities `P(Qc|Qr)/P(Qc)`, where `r` is the number of the row, and `c` the number of the column of the questions similarity matrix. I.e., the cell `(r,c)` of the matrix gives the conditional probability of answering question c correctly given that the question `r` was answered correctly. `P(Qc)` is the probability of answering question `c` correctly unconditionally. We see, e.g., that students answering `Q3` correctly had a higher chance of answering `Q4` correctly: 25% vs 15%. Of course, low numbers indicate also low support of the rule, hence few students supporting it. In general, answering `Q3` correctly has an influence on almost all following questions; answering `Q6` correctly has a big influence on `Q10`: from 68% to 100%.

However, due the curse of dimensionality, and poor representation spaces (redundant and correlated attributes, many of which may contain little discriminative/descriptive information) the performance of the many data-mining techniques seriously deteriorates. Another serious problem worth mentioning here is the highly unbalanced distribution of the instances in the dataset along some of the key dimensions (such as learning style and feedback requests) caused by the uncontrolled nature of the real assessment of the students (so that, e.g., sampling students into different groups is not ethical when they compete for a grade). We could hardly get any meaningful outcome as for finding groups of students with respect to their performance or feedback preferences in a fully automated manner. Nevertheless, association analysis was of some help for particular tasks.

4.3 Association analysis

Association rules express associations between different attributes. An association rule `X=>Y` expresses that the occurrence of `X` has a positive influence on the occurrence of `Y`. Whenever `X` is valid, the probability that `Y` appears is high.

Despite of a popular belief that association analysis often finds something useful and interesting from real datasets, we would like to stress here that in fact real success is fairly rare due to several reasons. In particular, many studies of association rules share one common problem: the output of association and pattern mining is often very large, in the sense that sometimes there are even more patterns than there were data to begin with. Also, the output contains often many redundant patterns, or is even misleading. We also experienced this even with our modest-size dataset, when the rules that define grade as a combination of response correctness and response certitude may not appear in the top *n* rules of the result set (unless we leave just these three sets of attributes) due to the presence of many other (hundreds) inherently redundant patterns.

Therefore, we switched to the semiautomatic discovery of the association rules focusing the search by predefining what should appear on the right hand side and what may appear on the left hand side of a rule. In particular, we were interested to find out what behavior

may explain failing or passing the test, how feedback preferences differ for the students with different leaning styles and for similar questions. Association rules discovered in this way helped to find out, for example, that when answering a question correctly and reading example-based EF students often did not request additional theory-based EF; and we found that students with reflective and sequential learning styles who failed the test often did not study any EF. However, it is worth mentioning that the effectiveness of this approach is still rather questionable for this type of assessment data since applying focused (yet "manual") SQL queries and performing statistical analysis of the data might still require less effort than "automatic" association analysis. Certainly, DM approaches still can be potentially much more effective and therefore, favorable.

5 Conclusions and Future work

In this paper we described an EDM case study based on online assessment data where students were able to receive tailored immediate EF after answering each of the questions in the test. Our case study was aimed towards showing that even with a modest size dataset and well-defined problem(s), for researchers having some prior experience in theory and practice of DM it is still rather hard to obtain meaningful results with a set of traditional DM techniques including clustering, classification and association analysis. This outcome calls for further research in the directions of (1) defining appropriate interestingness measures for the patterns to be mined, (2) the integration of prior domain knowledge (not necessarily subject domain, but also knowledge about the adaptation rules implemented in particular WBLS components) into the DM techniques, and (3) tailoring DM technology towards the EDM needs in general.

Our further work in this direction will be devoted to the development of a generic EDM framework that is able to (re)discover background knowledge and incorporate this knowledge into the mining process focusing it on the discovery of the truly interesting patterns, and in particular, the identification of subgroups and emerging patterns.

References

[1] Hatie, J., Timperley, H. The power of feedback, *J. Review of Educational Research*, 87 (1), 2007, p. 81–112.

[2] Mory, E. Feedback research revisited, In: *Jonassen, D. (eds.) Handbook of research on educational communications and technology*. Mahwah, NJ, 2004, p. 745–783.

[3] Vasilyeva, E., De Bra, P., Pechenizkiy, M., Puuronen, S. Tailoring feedback in online assessment: influence of learning styles on the feedback preferences and elaborated feedback effectiveness. *Proceeding of 8th IEEE ICALT 2008*.

[4] Vasilyeva, E., Pechenizkiy, M., De Bra, P. Adaptation of elaborated feedback in e-learning. *Proceeding of 5th AH 2008*.

[5] Witten, I., Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*, Second Edition, 2005, Morgan Kaufmann.

Machine Classification of Peer Comments in Physics

Kwangsu Cho
chokw@missouri.edu
School of Information Science and Learning Technologies
University of Missouri, Columbia

Abstract. As part of an ongoing project where SWoRD, a Web-based reciprocal peer review system, is used to support disciplinary writing, this study reports machine learning classifications of student comments on peer writing collected in the SWoRD system. The student comments on technical lab reports were first manually decomposed and coded as praise, criticism, problem detection, solution suggestion, summary, or off-task. Then TagHelper 2.0 was used to classify the codes, using three frequently used algorithms: Naïve Bayes, Support Vector Machine, and a Decision Tree. It was found that Support Vector machine performed best in terms of Cohen's Kappa.

1 Introduction

Writing is known as an important tool for learning and communication in science and engineering. However still undergraduate students have less than enough writing opportunities to learn to write. Instructors are simply overwhelmed by the workload of grading and commenting on writing assignments, and therefore tend to avoid administrating writing assignments in their courses.

As an alternative to the current expert-centric approach, peer reviewing is becoming very popular to improve this unfortunate *writing crisis* in the U.S. [1]. Peer collaboration strategy can be extremely valuable to producing more writing opportunities in the class, while it does not increase the existing workload of instructors. In addition, peer reviewing can be reliable for evaluation and effective for improving the given draft [1, 4].

Although peers are practically a good source of feedback, the helpfulness of their comments on peer writing can be very limited. One of the reasons is that undergraduate students in science and engineering are very likely to be novices. In addition, they tend to be inexperienced in writing and generating constructive comments. Therefore, it is critical for the successful use of peer reviewing that students come to generate helpful reviews for their peer writers in the absence of strong expert input. Also, reciprocal peer reviewing of writing is hard for instructor to monitor. As the class size increases, the amount of peer comments increases in an exponential way.

To address the peer review problems, in this study, I present a machine learning system that classifies student written comments as *helpful* or *not helpful*. The application takes student written comments as input and produces a classification as output. If a student' comments are classified as *not helpful*, the student reviewer is asked to revise the comments by the machine learning classifier. I report one of the three classification techniques in the system.

2 Comment Categories

There has been proliferation in feedback research. Although feedback is considered to be helpful for receivers, [6] found that 40% of the feedback intervention research they reviewed in fact reveled no positive impact or a negative impact on performance. Thus it is important to find what kinds of feedback would be helpful. Past research shows that two important features of helpful feedback are specificity and praise [2].

2.1 Specific vs. Non-specific Comments

One of the most important factors of helpful comments is specificity. Following Hattie and Timerley [5], helpful feedback should address three main questions: "Where am I going? (What are the goals?), How am I going (What progress is being made toward the goal?), and Where to next? (What activities need to be undertaken to make better progress?)" (p. 86). Therefore, helpful comments include what is a problem that an author should work on [problem detection) and also how the problem could be improved (solution suggestion) [2]. Straub [9] found that students prefer specific comments that provide explicit suggestions rather than vague comments and negative comments. With instructor comments, it was found that students are less likely to improve their writing if the comments are ambiguous. Consistently, Conrad and Goldstein [10] found that specific feedback from teachers tend to support student revision strategies.

2.2 Praise vs. Criticism Comments

Consistent with prior work showing the valued nature of praise and mitigating language in feedback, peers generally rated praise as being very helpful [2]. Some social psychologists [8] have argued that positive evaluation information is more likely to be accepted than negative information. However, affective tone is regarded as not directly improving peer writing because the tonal information does not tend to include much information about how to improve a detected problem [5]. Consistently, Kluger and DeNisi's [6] meta-analysis found praise feedback has a very small effect size. Unlike oral comments, written comments tend to make tonal information less distinct. However tonal information in written comments may still affect student writing. Gee [4] found that high school juniors who never received praise turned in shorter final drafts. Seideidman [7] found that high school students who consistently received positive feedback across eight writing assignments produced more optional rough drafts and revisions than either those who received negative comments or no comments.

3 Experiments

3.1 *Written Comment Data Collection*

Participants. This study used written comments collected from 44 undergraduate students in an intro-level Physics course who participated as a part of their course requirements. Individual students played two roles, one of writer and one of reviewer. Each student was asked to write first and revised drafts of technical research papers. In addition, each student reviewer was randomly assigned four peer papers. The reviews

were double-blinded: authors had pseudonyms and reviewers merely were identified as numbers to the authors. Their age and race information was not collected. The reciprocal peer reviewing activities were supported by the SWoRD system [1].

Writing and Reviewing Tasks. Individual students as writers wrote and revised technical lab reports about sound and human hearing. The reports consisted of title, abstract, introduction and theory, experimental setup, data analysis, conclusion, and references. To motivate students to generate a best possible quality of first drafts, they were instructed that first and revised/final drafts would be weighted equally; that is, first and final drafts would contribute equally to the final grade. Individual students as reviewers evaluated four peer drafts that were randomly selected by the SWoRD system with using a moving window algorithm. Reviewers and writers were blind to each other. Reviewers were given one week to read peer drafts and generate reviews on them. Reviewers were required to evaluate each draft both qualitatively and quantitatively along with three dimensions. For each dimension, they wrote comments and then provided a rating along a seven-point rating scale from disastrous (1) to excellent (7).

Coding Scheme of Peer Comments. All the 612 sets of written comments generated by the 44 reviewers were coded. First, each comment was decomposed into idea units which were defined as a self-contained message on a single piece of strength or weakness found in peer writing. Then, each unit was assigned to one of praise, criticism, problem detection, solution suggestion, summary, or off-task categories as discussed above. It should be noted that we rarely found criticism, summary, or off-task categories. A second coder independently evaluated 10% of randomly selected written comments. First the agreement between the two coders on segmentation was examined as the number of matched segmentations and categorizations divided by the number of total segments. For the segmentation, the coders reached 98.8 % agreement and for the categorization, they reached 98.6% agreement.

Written Comment Data Collection Procedure. All the students followed the built-in SWoRD procedures. After the instructor set the course with several parameters including the number of papers each student had to write, the number of peer reviews each paper would receive (and thus how many reviews each student had to complete), the students wrote two papers (via two drafts), each draft paper received reviews from four peer reviewers, and the students were given one week for each phase, although this research focused on the first paper. Reviewers were required to evaluate each draft, both qualitatively and quantitatively, along three dimensions, (1) Introduction, theory and experimental setup, (2) Data analysis and result, and (3) Abstract and conclusion. For each dimension, they wrote comments and then provided a rating along a seven-point rating scale from *disastrous (1)* to *excellent (7)*. A rubric guided the rating task and guidelines structured the commenting-giving task.

3.2 Machine Classification Techniques of Helpful Comments

While much classification research focused on semantics, a relatively small number of studies examined non-semantic information classification in text. For the comments classification, TagHelper 2.0 [3] was used. Compared to many other text classification

tools, TageHelper is very convenient, including a wide range of configurable text classification algorithms. To prepare the input comments, first the student comments were decomposed into idea units which were defined as a self-contained message on a single piece of strength or weakness found in peer writing. There were 2,441 idea units. Then, each unit was coded as praise, criticism, problem detection, solution suggestion, summary, or off-task comment. Second, among the total 2241 units after removing 200 units coded for summary and off-task instances were randomly split into two data sets: a training set of 1120 comment units and a test set of 1121 comment units. To develop a training model, selected were punctuation removal, unigrams, bigrams, Part-Of-Speech, line length, and rare feature removal in TagHelper. Three algorithms were used to compare its performance: Naïve Bayes, Support Vector Machines (SMO in TagHelper), and a decision tree (J48 in TagHelper). Finally, the test was used to test the performance of the models.

4 Results and Discussion

Table 1. Performance of Classification Approaches

	Training (n=1120)			Test (n=1121)		
	Naïve Bays	SVM	Decision Tree	Naïve Bays	SVM	Decision Tree
Correctly classified instances	782 (69.8%)	813 (72.6%)	729 (65.1%)	784 (66.8%)	782 (60.0%)	699 (62.4%)
Incorrectly classified instances	338 (30.2%)	307 (27.4%)	391 (34.9%)	372 (33.2%)	338 (30.0%)	421 (37.6%)
Cohen's Kappa	.52	.57	.44	.48	.54	.41

Table 1 shows the experimental results. With the training set, the highest performance measured as Cohen's Kappa was achieved with SVM. Although the performances of the models were a little decreased with the test dataset, the highest Cohen's Kappa was still found with SVM. This result is consistent with other text classification studies. To identify the source of errors or performance reduction, we analyzed the confusion matrices provided by TagHelper. All the three approaches revealed consistent problems: Praise comments were correctly categorized (80% correct vs. 20% incorrect for example in SVM) while problem detection comments tended to be confused with solution suggestion. Interestingly, problem detection was categorized as solution suggestion more than solution suggestion was categorized as problem detection.

5 Conclusion

This study has presented machine learning technologies applied for classifying peer comments in writing. As demonstrated with TagHelper, the machine learning technologies were found to be useful to categorize student comments on peer writing. Especially SVM achieved a noteworthy performance. Another important result was that the machine learning technologies, especially SVM, was good at categorizing tonal information into praise vs. non-praise. Finally, it should be noted that a hidden benefit of

the text classification technologies seems to help researchers develop coding schemes precisely.

Obviously, one of the benefits in using text classification technologies is automatically categorizing a large corpus of peer comments. This may be important in reciprocal peer reviewing of writing. Along with the class size, students tend to exchange an exponential amount of comments. Thus, automatic corpus coding technologies may be greatly helpful to help instructors to monitor reciprocal peer reviewing of writing in their classes. In addition, the current findings have been used to develop an intervention for student reviewers. Thus, based on the helpful comment model developed with TagHelper, student comments can be classified online before they are passed to their authors in order to help students generate constructive comments.

References

[1] Cho, K., & Schunn, C. D. Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Computers & Education*, 2007, 48(3), 409-426.

[2] Cho, K., Schunn, C., & Charney, D. Commenting on writing: Typology and perceived helpfulness of comments from novice peer reviewers and subject matter experts. *Written Communication*, 2006, 23, 260-294.

[3] Donmez, P., Rose, C. P., Stegmann, K., Weinberger, A., and Fischer, F. Supporting CSCL with automatic corpus analysis technology. *Proceedings of Computer Supported Collaborative Learning*, 2005.

[4] Gee, T. Students' Responses to Teachers' Comments. Research in the Teaching of English, 1972, 6, 212-221.

[5] Hattie, J., & Timperley, H. The power of feedback. *Review of Educational Research,* 2007, *77*(1), 81-112.

[6] Kluger, A. N., & DeNisi, A. The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin*, 1996, *119*(2), 254-284.

[7] Seidman, E. Marking Students' Compositions: Implications of Achievement Motivation theory. *Dissertation Abstracts International*, 1968, 28, 2605-A.

[8] Shrauger, J. S., & Schoenemann, T. J. Symbolic interactionist view of self-concept: Through the looking glass darkly. *Psychological Bulletin*, 1979, 86, 3, 549-573.

[9] Straub,R. Students' Reactions to Teacher Comments: An exploratory study. *Research in the Teaching of English*, 1997, 31, 91-119.

[10] Conrad, S. M., & Goldstein, L. M. ESL student revision after teacher-written comments: Text, contexts, and individuals. *Journal of Second Language Writing,* 1999, *8*(2), 147-179.

A pilot study on logic proof tutoring using hints generated from historical student data

Tiffany Barnes[1], John Stamper[1], Lorrie Lehman[1], and Marvin Croy[2]
{tbarnes2, jcstampe, ljlehman, mjcroy}@uncc.edu
[1]Computer Science Department, [2]Philosophy Department,
University of North Carolina at Charlotte

Abstract. We have proposed a novel application of Markov decision processes (MDPs), a reinforcement learning technique, to automatically generate hints using historical student data. Using this technique, we have modified a an existing, non-adaptive logic proof tutor called Deep Thought with a Hint Factory that provides hints on the next step a student might take. This paper presents the results of our pilot study using Deep Thought with the Hint Factory, which demonstrate that hints generated from historical data can support students in writing logic proofs.

1 Introduction and Related Work

Our goal is to automate the creation of intelligent tutoruing systems (ITSs) using educational data mining. We present experimental results of using the Hint Factory, a novel technique that uses a Markov Decision Process (MDP), created automatically from past student data, to generate specific contextualized hints to transform existing computer aided instruction (CAI) tools into intelligent tutoring systems (ITSs).

Example-based authoring tools such as CTAT use demonstrated examples to learn ITS production rules [3]. In these tools, teachers work problems in what they predict to be frequent correct and incorrect approaches, and then annotate the learned rules with appropriate hints and feedback. This system has also been used with data to build initial models for an ITS, in an approach called Bootstrapping Novice Data (BND) [4]. However, in both of these approaches, considerable time must be spent in identifying student approaches and creating appropriate hints.

In the Logic-ITA tutor, educational data mining was used to create hints that warn students of likely mistakes [6]. In contrast, we use a domain-independent method to build student models from data and automatically generate contextual, goal-oriented hints. In our prior work, we have visualized problem attempts to find students' areas of difficulty [8] and predicted availability and specificity of hints [1]. In this work, we present our experimental study of automated hint generation in the Deep Thought tutor using educational data mining.

2 The Hint Factory

The Hint Factory consists of the MDP generator and the hint provider. The MDP generator is an offline process, but the hint provider must be integrated with the CAI. In this experiment we modified the deductive logic Deep Thought tutor to provide hints while students work problems. The modifications needed were minimal, including tracking the student actions, passing them to the hint provider, and adding a hint button. Some work must be done to word the hints, but need only be done once.

The MDP Generator uses historical student data to generate a Markov Decision Process (MDP) that represents a student model, containing all previously seen problem states and student actions. Each action is annotated with a transition probability P and each state is assigned a value based on the MDP reward function R. On executing action a in state s the probability of transitioning to state s' is P(s' | s, a) and the expected reward associated with that transition is R(s'| s, a). Our method takes the current premises and the conclusion as the state, and the student's input as the action. Therefore, each proof attempt can be seen as a graph with a sequence of states (each describing the solution up to the current point), connected by actions. Specifically, a state is represented by the list of premises generated in the student attempt, and actions are the axioms (rules) used at each step.

We combine all student solution graphs into a single graph. We then use reinforcement learning to find an optimal solution to the MDP. In this work, we set a large reward for the goal state (100) and penalties for incorrect states (10) and for each action (1). We apply the value iteration reinforcement learning technique using a Bellman backup to assign reward values to all states in the MDP. An iterative gradient algorithm is used to calculate state values until convergence. The reward values for each state then indicate proximity to the goal, while probabilities of each transition show the frequency of taking an action in a certain state. Using the MDP, we create a hint file for each problem in the Deep Thought tutor. The hint file consists of all problem states generated in the MDP with available hints, i.e. non-errors with paths to the goal. Table 1 shows the number of student attempts used to create the MDPs, including the average, minimum, maximum, and expert lengths for each problem. The problems are listed in order of difficulty.

Figure 1 shows the graphical interface for Deep Thought, a custom CAI for logic proofs [2]. Our new hint button appears, as shown at the lower right in Figure 1, when a student loads a problem with hints. The button is bright yellow to make it more visible. When the hint button is pressed, the hint provider searches for the current state and, if a successor state exists, the successor with the highest value is used to generate a hint sequence. We work with instructors to determine the wording and order of hints. However, these are easily changed, and experiments can verify the appropriateness and effectiveness of the chosen hints.

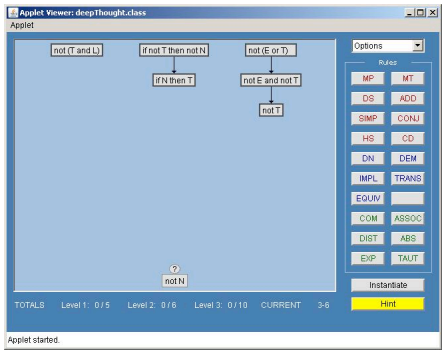


Figure 1: Problem 3.6 in Deep Thought.

Table 1: Average statistics for completed attempts used to create MDPs. Length includes errors; Expert reflects the length of the shortest expert proof.

Problem	3.6	3.8	3.2	3.5
Attempts	26	25	16	22
Length	8.0	11.9	11.3	18.8
Correct	5.5	11.2	9.0	16.7
Errors	1.1	1.1	0.9	3.1
Time	3:23	6:14	4:25	9:58
Expert	3	6	6	8

Our hints were designed based on instructor tutoring, research on hint strategies, and consistency with existing tutors. Both of our logic instructors prefer hints that help students set intermediate goals, as in [5]. We also use common pointing hints that

help focus attention and bottom-out hints that reveal the answer. For a given state, we generate a hint sequence of four parts as shown in Table 2. Each time the hint button is pressed, the next hint is given. Once a student performs a correct step, the hint sequence is reset. For each hint request, we record the time, hint sequence number, and the total number of hint requests.

Table 2: Hint sequence for problem in Figure 1, derived from example student data

Hint #	Hint Text	Hint Type
1	Try to derive not N working forward	Indicate goal expression
2	Highlight if not T then not N and not T to derive it	Indicate the premises to select
3	Click on the rule Modus Ponens (MP)	Indicate the rule to use
4	Highlight if not T then not N and not T and click on Modus Ponens (MP) to get not N	Bottom-out hint

4 Pilot Study

The main goal of this experiment was to test the capability of the Hint Factory to generate hints for actual students. Our secondary goal was exploratory, to determine how students used the provided hints, to inform our future work. Once the Hint Factory was added to Deep Thought, we generated MDPs and hint files for several Deep Thought problems. Since the current semester was already underway, we chose four level 3 problems from Deep Thought, 3.2, 3.5, 3.6, and 3.8. In case of unexpected errors, we enabled the instructor to quickly disable hints in Deep Thought. Fortunately, the software worked well and this was not necessary.

MDPs were generated using Deep Thought data from two 2007 Deductive Logic philosophy courses: spring (30 students) and summer (20 students). The data were cleaned by removing all incomplete proofs and log files with missing data. Forty students in the spring 2008 course were assigned to work these four problems.We hypothesized that, with hints, a higher percentage of students (by class and by attempts) would complete the proofs. Class participation and completion rates for the experimental class were much higher than the source class. For 2008, the attempt and completion rates were 88 and 83%, respectively, out of 40 students. For 2007, these rates were at most 48%, out of 50 students. This may be due to a novelty effect. Figure 2 shows the percent of solution attempts that are complete for the source (2007) and hints (2008) groups. For all problems but 3.5, there was a slightly higher percent complete with hints available. Problem 3.5 showed much higher completion rates for the hints group. Figure 3 shows a comparison in behavior during complete and partial solutions. Partial problems were longer by both time and the number of actions. Complete solutions have fewer errors and deletions and more hint usage.

Figure 4 shows the number of hints for each problem, broken down by color into the distribution of hint sequence length. We hypothesized that, as learning occurs, the usage of shorter hint sequences should increase. We do not have reliable data on the sequence of problems that students performed, so we have ordered problems by difficulty. We see here that students did use more hints as we move from less to more difficult problems, and the number and proportion of hint sequences of length 1 seems to go up from 3.6 to 3.8 and from 3.2 to 3.5. In our future work we will investigate the particular effects of using hints in transfer of learning.

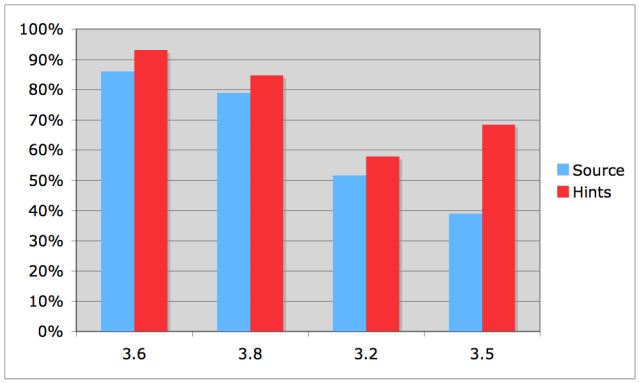


Figure 2: Percent attempt completion between the source and hints groups

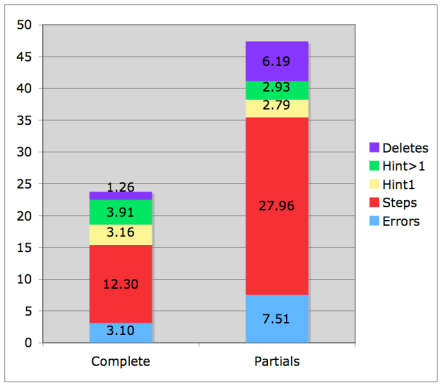


Figure 3: Comparison of behavior between complete and partial solutions

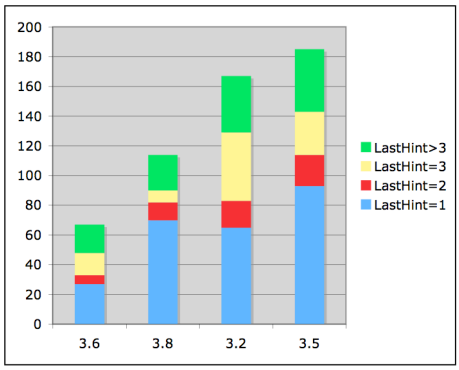


Figure 4: Distribution of hint sequences by seq. length

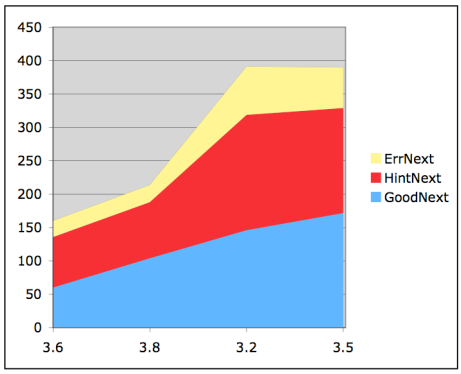


Figure 5: Number of steps after a hint that are correct (good), hints, or errors

Another way to measure the effectiveness of hints is to examine behavior just after receipt of a hint. We therefore investigated the number of errors, correct or good steps, and hint requests immediately following a hint, as shown in Figure 5. The proportion of good steps just after a hint goes consistently up, while there is a jump in the number hints and errors requested between problems 3.8 and 3.2. When we examine the difference between 3.2 and 3.5, we see more good steps, and slightly more hints and fewer errors just after a hint. Along with its higher completion rate, this suggests that the hints may be more effective for 3.5.

Table 4 shows the hint usage and availability for all 2008 completed and partial attempts. "Moves" is the total number of non-error student actions in the interface. In our prior feasibility study, we predicted that proof MDPs built using 16-26 attempts on problem 3.5 have a probability of providing hints 56-62% of the time [1]. In this experiment, if a student had pressed the hint button after every move taken, a hint would have been available about 48% of the time. This is lower than our prediction. However, the existence of the hint button may have changed student behavior.

We were encouraged by the comparison of this rate with the hint availability when students requested hints. Table 4 shows that a hint was delivered for 91% of hint requests. This suggests that hints are needed precisely where we have data in our MDPs from previous semesters. We plan to investigate the reasons for this surprising result with further data analysis and experiments. It is possible that there are a few key places where many students need help. Another explanation is that, when students

are performing actions that have not been taken in the past, they may have high confidence in these steps and need no help.

Table 4: Hint usage and availability by problem, including all solution attempts in Spring 2008

Problem	3.2	3.5	3.6	3.8	Total
Attempts	69	57	44	46	216
Moves	999	885	449	552	2885
Moves w/ Avail. Hints	442	405	230	269	1346
% Moves w/ Avail. Hints	**44.2%**	**45.8%**	**51.2%**	**48.7%**	**47.9%**
Hint1 Requests	236	232	70	154	692
Hint1 Delivered	213	212	66	142	633
% Hint1s Delivered	**90.3%**	**91.4%**	**94.3%**	**92.2%**	**91.5%**

5 Conclusions and Future Work

We have presented the implementation and pilot experimental results on our system to use MDPs to create student models and automatically generated hints. Our approach to creating intelligent support for learning differs from prior work in authoring tutoring systems by mining actual student data to provide goal-oriented contextual hints. We have demonstrated that our method provides hints after just one semester, and can be adapted to add new hints as more data are collected. In our future work, we plan to use machine learning to generate hints for problems with no data and to test our method in varied domains.

References

[1] Barnes, T. and Stamper, J. Toward automatic hint generation for logic proof tutoring using historical student data, To appear in *Proc. Intl. Conf. Intelligent Tutoring Systems (ITS2008),* Montreal, Canada, June 23-27, 2008.

[2] Croy, M. Problem solving, working backwards, and graphic proof representation, *Teaching Philosophy* 23 (2000), 169-187.

[3] Koedinger, K., Aleven, V., Heffernan. T., McLaren, B. & Hockenberry, M. Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. *Intelligent Tutoring Systems*, Maceio, Brazil, 2004, pp. 162-173.

[4] McLaren, B., Koedinger, K., Schneider, M., Harrer, A., & Bollen, L. Bootstrapping Novice Data: Semi-automated tutor authoring using student log files, In Proc. Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes, *Intelligent Tutoring Systems,* Maceió, Brazil, 2004.

[5] McKendree, J.: Effective Feedback Content for Tutoring Complex Skills. *Human-Computer Interaction* 5(4), pp. 381-413. Lawrence Erlbaum, 1990.

[6] Merceron, A. & Yacef, K.: Educational Data Mining: a Case Study. In *12th Intl. Conf. Artificial Intelligence in Education*, Amsterdam, Netherlands, IOS Press, 2005.

Argument Mining Using Highly Structured Argument Repertoire

Safia Abbas[1] and Hajime Sawamura[2]

[1] Graduate School of Science and Technology, Niigata University
8050, 2-cho, Ikarashi, Niigata, 950-2181 JAPAN
safia@cs.ie.niigata-u.ac.jp
[2] Institute of Natural Science and Technology,
Academic Assembly,Niigata University
8050, 2-cho, Ikarashi, Niigata, 950-2181 JAPAN
sawamura@ie.niigata-u.ac.jp

Abstract. Argumentation theory is considered an interdisciplinary research area. Its techniques and results have found a wide range of applications in both theoretical and practical branches of artificial intelligence, education, and computer science. Most of the work done in argumentation use the on-line textual data (i.e. unstructured or semi-structured) which is intractable to be processed. This paper reports a novel approach to build a Relational Argument DataBase (RADB) with managing tools for argument mining, the design of the RADB depends on the Argumentation Interchange Format Ontology(AIF) using "Walton Theory". The proposed structure aims to: (i) summon and provide a myriad of arguments at the user's fingertips, (ii) retrieve the most relevant results to the subject of search, (iii) support the fast interaction between the different mining techniques and the existing arguments, and (iv) facilitate the interoperability among various agents/humans.

1 Introduction

Argumentation theory, or argumentation, embraces the arts and sciences of civil debate, dialog, conversation, and persuasion. It studies rules of inference, logic, and procedural rules in both artificial and real world settings. Argumentation is concerned primarily with reaching conclusions through logical reasoning, that is, claims based on premises. Its techniques and results have found a wide range of applications in both theoretical and practical branches of artificial intelligence, education, and computer science [8, 7, 11]. Among other things, we are mainly concerned with argument mapping, analysis and formal computational argumentation frameworks, where many efforts have been most devoted as far as we see in the literature[9].

Argument mapping (e. g., Compendium, Araucaria, Rationale, etc.) aims at improving our ability to articulate, comprehend and communicate reasoning, by producing diagrams of reasoning and argumentation for especially complex arguments and debates. It is greatly anticipated that it helps students learning critical thinking methods as well as uses promoting critical thinking in the daily life. On the other hand, the main concern in formal computational argumentation frameworks is to formalize methods in which the final statuses of arguments are to be decided semantically and/or dialectically [8].

Argument mining and argument discovery technologies are particularly needed to summon the arguments, and help the user who is looking for specific subject or piece of information over a large-scale database. In this paper, we present a novel approach to sustain argument analysis, retrieval, and re-usage from a relational argument database (highly structured argument repertoire). Different mining techniques are used to provide a myriad of arguments at the user's fingertips, refine the user's analysis background, and reveal more relevant search results. This work mainly concerns with mining arguments through RADB, which summarizes the argument dataset. Such representation supports the fast interaction between the different mining techniques and the existing arguments, and facilitates the interoperability among various agents/humans. In addition, the framework outlines are illustrated, where a mining classifier agent is integrated with an intelligent tutoring system (ITS).

Such integration assists in deepening the understanding of negotiation, decision making, critical thinking and improving the analysis and intellectual process of students.

The paper is organized as follows. Section 2 illustrates the design and the implementation of the relational argument database (highly structured argument repertoire) together with different mining techniques. Section 3 motivates our work and compares it to the most relevant work performed in the field. Finally, conclusions and future work is presented in Section 4.

2 Relational Argument DataBase

A relational database can be defined as a set of information reformulated and categorized into a set of files (tables) that can be accessed, gathered (queried), and manipulated in different manners. According to the AIF ontology, arguments can be represented semantically in the form of nodes connected with directed edges in a directed graph known as argument network[1]. If the cyclic problem (the same information node (I-node) refines more than one scheme node (S-node)) is avoided, the arguments can semantically be represented as directed tree, which can be structured in the form of well-established relational database, and annotated as relational argument database (RADB).

2.1 Design and Implementation

This subsection describes the building blocks for the relational arguments data bases. We consider the AIF ontology [1, 6] with some restrictions (such that no edge emanates from I-node to I-node), and Walton schemes [4] for arguments analysis. Any argument scheme based on Walton theory can be represented as shown in *Fig.*1. which represents a general skeleton for the different schemes

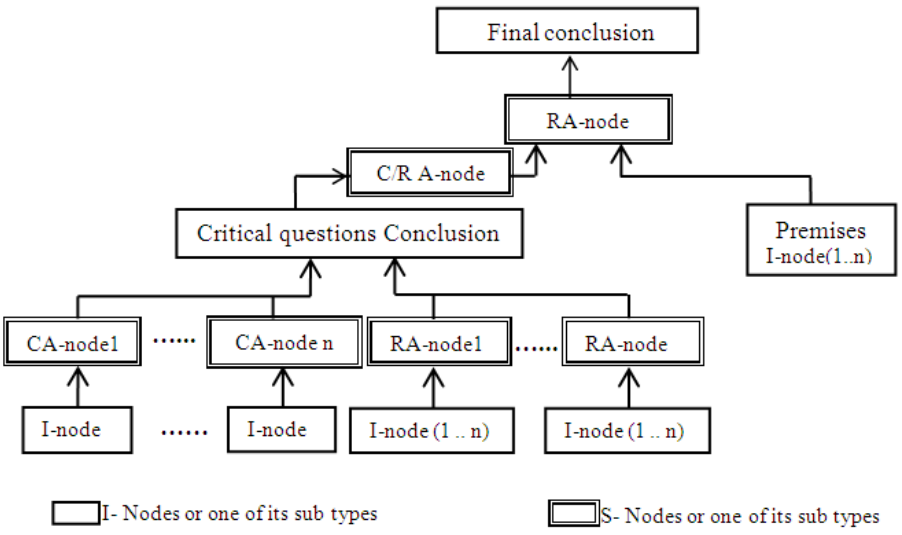
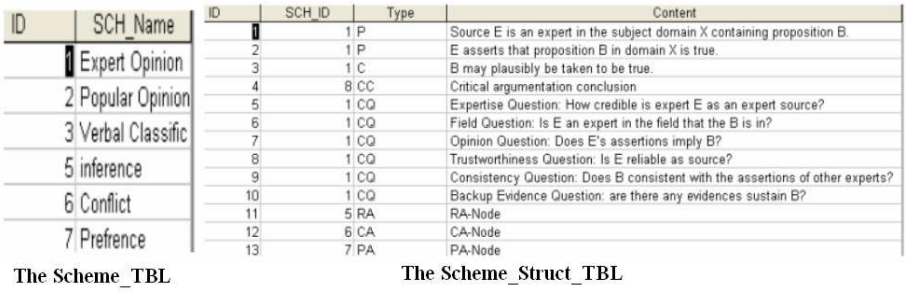


Fig. 1. Argument network representation for different Walton schemes

description in Walton theory[4]. The premises block gathers the different premises types (majors, minors). The critical question's conclusion block assembles the result of the different critical questions together with the results of the different presumption questions that are to be exposed in a specific scheme.

Considering the canonical representation for the schemes, we pose some sentiments about the relational database baselines. In our design, we gather the different scheme information into three basic files (tables): Scheme_TBL, Scheme_Struct_TBL and Data_TBL. First, the scheme kind is formulated in Scheme_TBL *Fig.* 2, in which rows act as records of data for different schemes, and columns as features (attributes) of records. The Scheme_TBL creates an identification number(ID) for each scheme name ($Scheme\_Name$), where this ID plays a role of primary key for the table and foreign key in the others. In addition, any ID attribute will stand for the same function in

all files. Scheme_Struct_TBL assembles the different information associated with different schemes. Such that, $Scheme\_Id$ stands for the foreign key of $Scheme\_TBL$, indicating the scheme concerned. The $Content$ field contains the details of the associated information (premises, conclusion,... etc.). The $Type$ field has four values, P for premises, C for conclusion, CQ for critical question and CC for critical argument conclusion. For instance, the expert opinion scheme[4] can be represented as shown in Scheme_Struct_TBL of $Fig.$ 2.



The Scheme_TBL

ID	SCH_Name
1	Expert Opinion
2	Popular Opinion
3	Verbal Classific
5	inference
6	Conflict
7	Prefrence

The Scheme_Struct_TBL

ID	SCH_ID	Type	Content
1	1	P	Source E is an expert in the subject domain X containing proposition B.
2	1	P	E asserts that proposition B in domain X is true.
3	1	C	B may plausibly be taken to be true.
4	8	CC	Critical argumentation conclusion
5	1	CQ	Expertise Question: How credible is expert E as an expert source?
6	1	CQ	Field Question: Is E an expert in the field that the B is in?
7	1	CQ	Opinion Question: Does E's assertions imply B?
8	1	CQ	Trustworthiness Question: Is E reliable as source?
9	1	CQ	Consistency Question: Does B consistent with the assertions of other experts?
10	1	CQ	Backup Evidence Question: are there any evidences sustain B?
11	5	RA	RA-Node
12	6	CA	CA-Node
13	7	PA	PA-Node

Fig. 2. The main structure of different schemes

The Data_TBL table contains all users' transactions. This table gathers all users' analysis for different arguments. The table consists of : the $Stru\_Id$ that serves as foreign key for the Scheme_Struct _TBL's ID, and refers to a specific part of the scheme details, the $Content$ attribute contains a portion of the analyzed text that fulfills the referred fixed scheme part, the $Type$ attribute, which holds three values only, 1 for the supported node, 0 for rebuttal node, -1 for undetermined value that denotes neither supported nor rebuttal node. Since we consider any argument network as a kind of directed rooted tree, the $Child\_Of$ attribute points to the parent of each node, whereas the root node has no parents (0 refers to no parent). The $level$ attribute refers to the level of each node in the tree, such that the value 0 indicates the root node. Finally, the $argumentation\_no$ attribute contains the number of the analyzed argument context.

For example, the following context below from Araucaria[2] repository database[2, 3, 12] is reanalyzed based on expert opinion scheme as shown in $Fig.$3 and $Fig.$4. *"Eight monthold Kyle Mutch's tragic death was not an accident and he suffered injuries consistent with a punch or a kick, a court heard yesterday. The baby, whose stepfather denies murder, was examined by pathologist Dr James Grieve shortly after his death. Dr. Grieve told the High Court at For far the youngest was covered in bruises and had suffered a crushed intestine as well as severe internal bleeding. When asked by Advocate Depute Mark Stewart, prosecuting, if the bruises could have been caused by an accident, he said "No. Not in a child that is not walking, not toddling and has not been in a motor car." Dr. Grieve said the injuries had happened "pretty quickly" and would be "difficult for an infant to cope with". The lecturer in forensic medicines at Aberdeen University told the jury that the bruises could have been caused by a single blow from a blunt instrument, like a closed hand. Death, not accident, court told, "Evening Telegraph", Monday, September 13, 2004, p.11"*

Regarding to the canonical representation for Waltons schemes presented in $Fig.$1, the given context could be analyzed as shown in $Fig.$3 based on expert opinion scheme [1]. Moreover, this analysis will be devolved through transaction records, as shown in $Fig.$4, to the structured data base (RADB) revealing the different parties of the analysis.

2.2 Framework Overview

Yun Chi et al. [13] surveyed the current algorithms used for mining frequent subtrees from databases. They focused on two main components of these algorithms, the candidate generation step and the support counting step. They revealed that there is no single best tree mining algorithm. Some algorithms offer a better time efficiency, while others require less memory. So every time we manipulate

[2] http:// araucaria.computing.dundee.ac.uk/

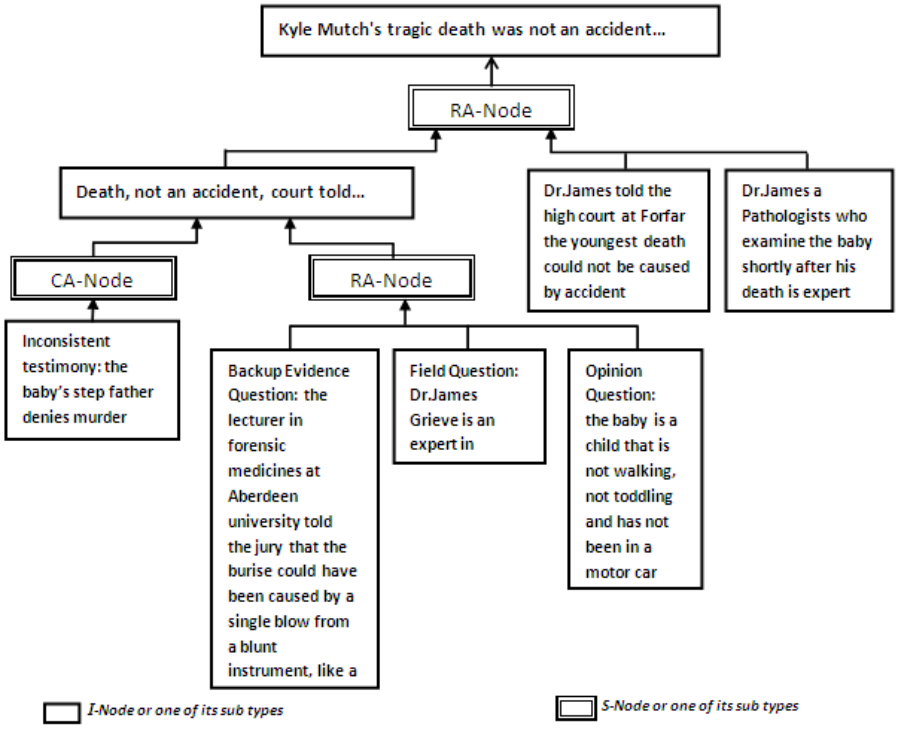


Fig. 3. The analysis diagram of the above context based on expert opinion scheme

ID	Stru_ID	Content	Type	Child_of	level	argumentation_no
1	3	Kyle Mutch's tragic death was not an accident and he suffered injuries	-1	0	0	argument_602
2	7	RA-Node	1	1	1	argument_602
3	2	Dr.James told the high court at Forfar the youngest death could not be cau	1	2	2	argument_602
4	1	Dr.James Grieveis a Pathologists who examne the baby shortly after his	1	2	2	argument_602
5	4	Death, not an accident, court told	1	2	2	argument_602
6	7	RA-Node	1	5	3	argument_602
7	6	Field Question: Dr.James Grieve is an exper in pathology	1	6	4	argument_602
8	7	Opinion Question: the baby is a child that is not walking,	1	6	4	argument_602
9	10	Backup Evidence Question: the lecturer in forensic medicines	1	6	4	argument_602
10	12	CA-Node	0	5	3	argument_602
11	9	conflict from inconsistent testimony: the baby's stepfather denies murder	0	10	4	argument_602

Fig. 4. The transaction records of the above analysis

the proposed RADB we will consider the time and memory consuming.

We draw a preliminary vision for retrieving and mining the RADB, using a framework with ITS component incorporated. The framework as depicted in $Fig.5$ consists of three main components: the *parser* module, the mining $classifier$ agent, and the ITS. The *parser* module receives a statement S from the intended users such as students or agents. the statement is divided by the parser into tokens, then the number of tokens is reduced. Finally the final crucial set of words { I_1, I_2,..., I_n } is sent to the $classifier$ agent. The tokens are reduced if they belong to a look up table containing the set of all unnecessary words like{a, an, the,...,etc }, otherwise it is added to the set of tokens to be sent to the $classifier$ agent. The importance of the *parser* module lies in reducing the set of tokens which in turn will reduce the number of iterations done by the classifier agent, and improve the complexity of the used mining algorithms.

The $classifier$ agent classifies the retrieved contexts depending on the students specification. The agent can classify the retrieved arguments by priority, polarity, scheme name, premises (with/against), and by conclusion. The priority aims to show the retrieved contexts organized by the maximum

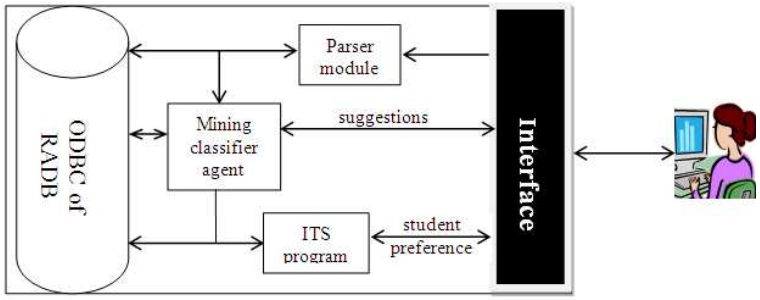


Fig. 5. Framework outline

support number based on the classification mining technique AprioriTid [10, 5]. Polarity classifies the retrieved arguments in to two classes, support class and against class, using the text mining techniques. Scheme name retrieves the desired contexts depending on a specific scheme name determined by the student. Premises (with/against) retrieves arguments by searching only in the different premises, and conclusion retrieves and classifies the arguments by searching only in the different conclusions. The classifier agent receives the set of crucial words { I_1,I_2, ..., I_n } from the parser module and the search type from the student, then retrieves and classifies the documents that are relevant to the student's search statement from the RADB using the multi-term text phrases approach[5] such that T={ T_1,T_2,...,T_m } is the collection of raw documents, I={ I_1,I_2,,I_n } is a set of words appearing in T. T'={ T'_1, T'_2,..., T'_m } is the set of documents, where T'_i contains a set of multi-term phrases I'={ I'_1, I'_2,..., I'_n }, $I'_i = I_j \bigcup I_{j+q} \bigcup .... \bigcup I_k$, where $1 \leq j \leq k \leq n$, $q \in$ [1,2,..,k-j], and I_i can appear in I'_i repeatedly. The importance of this classifier agent lays in manging the different mining techniques in order to: (i) direct the search towards hypotheses that are more relevant to the user's needs, (ii) add flexibility to the retrieving process to suit the users aims (iii) offer a myriad of arguments at users fingertips.

After the classifier agent exposed the pertinent contexts to the student, the student picks up one context among them. The student preference then delegates to the *ITS* program. The program exposes the corresponding context, and gives the student the ability to analyze the selected argument based on a specific chosen scheme. Finally the program negotiates with the student about the way of analysis through mining techniques to (i) provide constrains that guide the argument analysis process based on scheme structure and pre-existing arguments, (ii) refine the user's underlying classification, (iii) provide an analysis background to the different users, (iv) deepen the understanding of negotiation, decision making, (v) develop critical and intellectual thinking of students, and improve the analysis ability.

2.3 Illustrative Example

Suppose the student wants to know anything about Iraq war, so he/she come up with a statement "the destructive war in Iraq". First the parser module will divide the statement into tokens {the, destructive, war, in, Iraq}, such that I_1=the, I_2=destructive, I_3=war, I_4=in, I_5=Iraq, then access to the data base through the ODBC connection to compare each token with the lookup table entities and reduce the number of tokens, after checking the lookup table the tokens will be I_1=destructive, I_2=war, I_3=Iraq. So the output will be the item sets {destructive, war, Iraq}. Now the classifier should find the set of raw documents T= {T_1,T_2,...,T_n }. Assume the conclusion is the search criteria, so the classifier will use the mining AprioriTid algorithm [10, 5] to make all the possible combination of the item sets and classify the result depending on the support number for each combination. Firstly, the algorithm will calculate the support number for each single token, and select the tokens that have support number greater than *minsup*, that is a number specified by the student, however in our case we will take the *minsup*=1 so any token appears at least once will be considered. Since we assume that the student choose to search by conclusion then the support number for each token can be counted by the number of transactions resulted from the following select statements.

Select argument_no from Data_TBL where
Stru_id = 3 *and Content like '%destructive%';*

Select argument_no from Data_TBL where
Stru_id = 3 and Content like '%war%';
Select argument_no from Data_TBL where
struc_id = 3 and Content like '%Iraq%';

The out put of this step will be the set of ordered pairs L_1 = {(destructive, 5), (war, 10), (Iraq, 20)}, where the ordered pair is of the form (the token, the support number), and the set A_1={argument_801, argument_509,...} which contains the non repetitive arguments (argument_no) that contains these tokens. Secondly, the algorithm consequently builds the super set C_k = apriori_gen(L_{k-1}) for all possible combinations of the tokens. *Fig.*6. Shows the first iteration for C_2=apriori_gen(L_1).

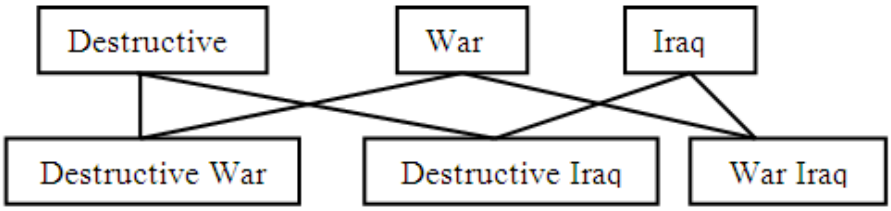


Fig. 6. The super set C_2 of the singleton token set L_1

Then the support number for each combination is checked through the set A_1. Suppose that the support number for the item set "War Iraq" is 0, which is less than the minsup=1, so this item set is neglected. The output of this iteration will be L_2= {(Destructive war, 3), (Destructive Iraq, 5)}, and the set A_2={argument_509,...}. Finally, the last iteration of our example will out put the set L_3={(Destructive war Iraq, 1)} and the set of arguments A_3={ argument_509}. Suppose that A_3 had more than one argument_no, we add function to the algorithm to check the counter argument of each argument_no and order the arguments depending on the possessed counter arguments, such that the argument that contains more cons is the weakest.

Therefore, the conclusions corresponding to the retrieved arguments are organized, such that the $argument_1$ is highly relevant to the issue of search rather than $argument_n$ as shown in *Fig.* 7. When the student pickup one of the classified output conclusions the ITS will access to the database to retrieve the corresponding context and then the context is exposed to the student giving him/her the ability to analyze. Furthermore, the ITS will negotiate with the student partially (step by step hints)

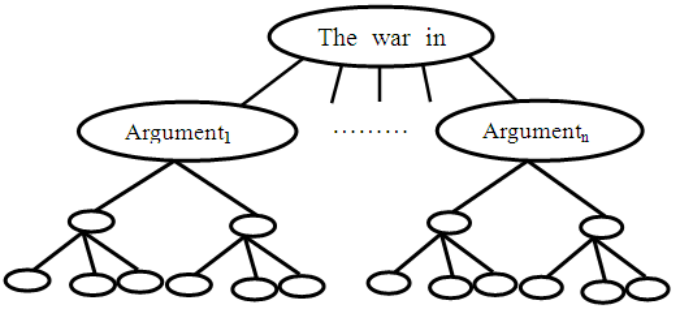


Fig. 7. The argument retrieval output

or totally (compare the student whole analysis with the original one retrieved from the repository) as discussed in the next section, in order to improve his/her analysis skill.

3 Motivation

In this paper, firstly we introduce a novel approach to retrieve the information using mining techniques based on RADB, which is a highly structured repertoire gathers the argument dataset, such that all needed information is encoded in an appropriate form. This structure facilitates fast interaction, and enjoys general applicability since it does not require a specialized knowledge. The idea is to mine the pre-existing arguments in order to (i) direct the search towards hypotheses that are more relevant to the users needs, even with more than one word in the search statement, (ii) add flexibility to the retrieving process to suit the users aims (iii) offer a myriad of arguments at users fingertips (iv) provide an analysis background to the different users. Secondly, we assemble the different retrieving techniques in a *Classifier agent* to be merged with an ITS. The agent based intelligent tutoring system aims to (i) provide constrains to guide the argument analysis process based on scheme structure and pre-existing arguments, (ii) refine the users' underlying classification, (iii) deepen the understanding of negotiation, decision making, develop critical and intellectual thinking of students, and improve the analysis ability.

I. Rahwan presents the ArgDf system [1, 6], through which users can create, manipulate, and query arguments using different argumentation schemes. Comparing ArgDf system to our approach, both of them sustain creating new arguments based on existing argument schemes. The available argument schemes are listed, enabling the user to choose the scheme to which the argument belongs. Details of the selected argumentation scheme are then retrieved from the repository, and the generic form of the argument is displayed to the user to guide the creation of the premises and conclusion. For example, querying the ArgDF repository to extract the name of the schemes can be done through the following RQL query:

$SelectScheme,\ PresumptiveInferenceScheme - hasSchemeName$
$FromScheme\ :\ kb : PresumptiveInferenceScheme\ kb : hasSchemeName$
$PresumptiveInferenceScheme - hasSchemeNameusingnamespace$
$rdf\ =\ http://www.w3.org/1999/02/22 - rdf - syntax - ns\#,$
$rdfs = http://www.w3.org/2000/01/rdf - schema\#,$
$kb = http://protege.stanford.edu/kb\#.$

whereas, in our approach, querying the RADB to extract the name of the schemes is done through the following SQL query:

$SELECT\ SCH\_Name,\ ID\ FROM\ [Scheme\_TBL]$

Consequently, to extract the details of the selected scheme, the following SQL query is performed:

$SELECT\ Content\ FROM\ [Scheme\_Struct\_TBL]\ WHERE$
$Id\_of\_sel\_scheme =\ [Scheme\_Struct\_TBL].SCH\_ID.$

In addition, the ArgDf system guides the user during the creation process based on the scheme structure only, the user relies on his efforts and his background to analyze the argument. However, in our approach, the user is not only guided by the scheme structure but also by crucial hints devolved through mining techniques. Accordingly, the creation process is restricted by comparing the contrasting reconstruction of the user's analysis and the pre-existing one. such restriction helps in refining the user's underlying classification.

In the ArgDf system, searching existing arguments is revealed by specifying text in the premises or the conclusion, as well as the type of relationship between them. Then the user can choose to filter arguments based on a specific scheme. Whereas, in our approach, searching the existing arguments is not only done by specifying text in the premises or the conclusion but also by providing different strategies based on different mining techniques (as explained in subsection 2.2). This method guarantees the retrieval of the most convenient hypotheses relevant to the subject of search.

4 Conclusions and Future Work

In this paper, we present a novel approach of building a highly structured argument repertoire (RADB) that uses different mining techniques to support argument analysis, retrieval, and re-usage. The paper also introduced an educational framework that utilizes the RABD. The proposed structure aims to: (i) summon and provide a myriad of arguments at the user's fingertips, (ii) retrieve the most relevant results to the subject of search, (iii) support the fast interaction between the different mining techniques and the existing arguments, and (iv) facilitate the interoperability among various agents/humans. Our attempt enjoys certain advantages when compared to others, especially with respect to the search of pre-existing arguments. The results obtained are very promising, where highly relevant and convenient arguments are obtained, especially when the search statement is in this form: "the destructive war in Iraq". Future work mainly concerns with the implementation of the rest of the framework components.

References

1. S. Modgil I. Rahawan C. Reed et.al. C. Chesnevar, J. McGinnis. Towards an argument interchange format. In *The Knowledge Engineering Review*, volume Vol. 00:0, pages 1–25. Cambridge University Press, 2007.
2. et.al. D. Walton, G. Rowe. Araucaria as a tool for diagramming arguments in teaching and studying philosophy. In *Teaching Philosophy*, volume Vol. 29, pages 111–124, 2006.
3. C. Reed G. Rowe and J. katzav. Araucaria: Making up argument. In *European Conference on Computing and Philosophy*, 2003.
4. M. Godden and D. Walton. Argument from expert opinion as legal evidence: Critical questions and admissibility criteria of expert testimony in the american legal system. In *Ratio Juris*, volume Vol 19, pages 261–286, 2006.
5. H.Ahonen-Myka. finding all maximal frequent sequences in text. In *Proceeding of ICML99 workshop*, 1999.
6. F. Zablith I. Rahawan and C. Reed. The foundation for a world wide argument web. In *Artificial Intelligence Conference (AAAI)*. published in the Artificial Intelligence Journal, April 04, 2007.
7. M. Baker J. Andriessen and D. Suthers. Arguing to learn confronting cognitions in computer-supported collaborative learning environments. In *Kluwer Academic Publishers, Dordrecht/Boston/London*, volume Vol.1, 2003.
8. C. Reed J. Katzav and G. Rowe. Argument research corpus. In *M.-P. Huget (ed.), Communication in Multiagent Systems,Lecture Notes in Computer Science,Springer Verlag, Berlin, Germany*, volume Vol.2650, pages pp. 269–283, 2003.
9. H. Prakken and G. Vreeswijk. Logical systems for defeasible argumentation. In *D. Gabbay and F. Guenther, editors, Handbook of Philosophical Logic*, pages 219–318. Kluwer, 2002.
10. R. Srikant R. Agrawal. Fast algorithms for mining rules. In *Proceeding of the 20th VLDB Conference Santiago, Chile*, 1994.
11. I. Rahawan and P.V. Sakeer. Representing and querying arguments on semantic web. In *Computational Models of Argument, P.E. Dunne and T.J.M. Bench-Capon (Eds.), IOS Press*, 2006.
12. C. Reed and G. Rowe. Araucaria: Software for argument analysis, diagramming and representation. In *International Journal on Artificial Intelligence Tools*, volume Vol.13, page pp.983, 2004.
13. R.R. Muntz. et.al. Y. Chi. Frequent subtree mining-an overview. In *Fundamenta Informaticae*, pages pp.1001–1038, 2001.

Skill Set Profile Clustering Based on Student Capability Vectors Computed From Online Tutoring Data

Elizabeth Ayers[1], Rebecca Nugent[1], and Nema Dean[2]
{eayers, rnugent}@stat.cmu.edu, {nema}@stats.gla.ac.uk
[1]Department of Statistics, Carnegie Mellon University
[2]Department of Statistics, University of Glasgow

Abstract. In educational research, a fundamental goal is identifying which skills students have mastered, which skills they have not, and which skills they are in the process of mastering. As the number of examinees, items, and skills increases, the estimation of even simple cognitive diagnosis models becomes difficult. To address this, we introduce a capability matrix showing for each skill the proportion correct on all items tried by each student involving that skill. We apply variations of common clustering methods to this matrix and discuss conditioning on sparse subspaces. We demonstrate the feasibility and scalability of our method on several simulated datasets and illustrate the difficulties inherent in real data using a subset of online mathematics tutor data. We also comment on the interpretability and application of the results for teachers.

1 Introduction

In educational research, a fundamental goal is identifying which skills students have mastered, which skills they have not, and which skills they are in the process of mastering. A variety of cognitive diagnosis models [10] address this problem using information from a student response matrix and an expert-elicited assignment matrix of the skills required for each item. However, even simple models [8] become more difficult to estimate as the number of skills, items, and examinees grows [1]. Any procedure used should be able to handle the missing values that arise in assessment; students may not have time to finish all items, for example, or they might intentionally skip items. Moreover, data is often missing by design in assessment embedded in online tutoring systems.

In response, we introduce a *capability matrix* showing for each skill the proportion correct on all items tried by each student involving that skill, expanding on the sum-score work of [6]. We apply clustering methods to the capability matrix to identify groups of students with similar skill set profiles, similar to [12] which clusters students based on their collaborative behavior. In addition, we propose a conditional clustering heuristic that takes advantage of obvious group separation in one or more dimensions. These methods are faster than common cognitive diagnosis models, provide a unique visualization tool of students' skill mastery, and scale well to large datasets. We show that these methods also add flexibility in the assignment of skill mastery; we are also able to determine the students' skills for which mastery is uncertain, a conservative classification scheme that does not force a hard skill mastery assignment of yes or no. For illustrative purposes, we demonstrate our method on three datasets simulated from the DINA model [8], a common cognitive diagnosis model, and on a small subset of data obtained from the ongoing IES Assistment Project [5]. Finally we conclude with comments on current and future work.

2 The Capability Matrix

We begin by assembling the skill dependencies of each item into a Q-matrix [2,13]. The Q-matrix, also referred to as a transfer model or skill coding, is a $J \times K$ matrix where $q_{jk} = 1$ if item j requires skill k and 0 if it does not, J is the total number of items, and K is the total number of skills. The Q-matrix is usually an expert-elicited assignment matrix. This paper assumes the given Q-matrix is known and correct.

$$Q = \begin{bmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,K} \\ \vdots & \ddots & & \vdots \\ q_{J,1} & q_{J,2} & \cdots & q_{J,K} \end{bmatrix}, \quad Y = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,J} \\ \vdots & \ddots & & \vdots \\ y_{N,1} & y_{N,2} & \cdots & y_{N,J} \end{bmatrix}$$

Student responses are assembled in a $N \times J$ response matrix Y where y_{ij} indicates both if student i attempted item j and whether or not they answered item j correctly and N is the total number of students. If student i did not answer item j then $y_{ij} = NA$. The indicator $I_{y_{ij} \neq NA} = 0$ expresses this missing value. If student i attempted item j ($I_{y_{ij} \neq NA} = 1$), then $y_{ij} = 1$ if they answered correctly, or 0 if they answered incorrectly.

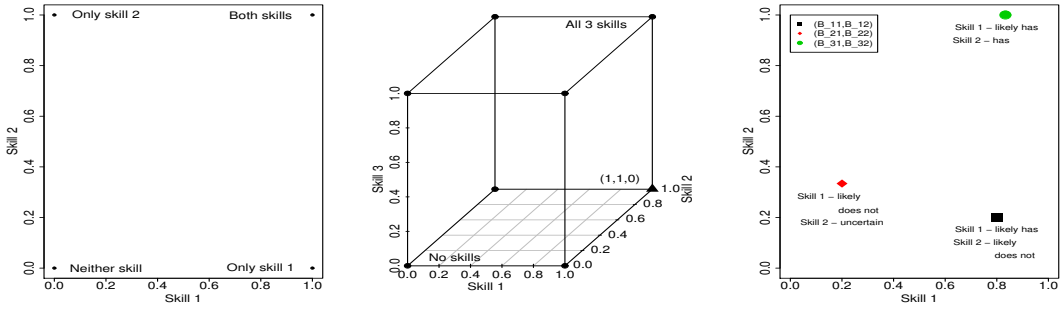
To cluster students by their skill set profiles, we need a summary statistic of their skill performance. We define an $N \times K$ *capability matrix* B, where B_{ik} is the proportion of correctly answered items involving skill k that student i attempted. If student i did not attempt any items with skill k, we assign a value of 0.5, an uninformative probability of skill mastery. That is, if $\sum_{j=1}^{J} I_{Y_{ij} \neq NA} \cdot q_{jk} = 0$, $B_{ik} = 0.5$. Otherwise,

$$B_{ik} = \frac{\sum_{j=1}^{J} I_{Y_{ij} \neq NA} \cdot Y_{ij} \cdot q_{jk}}{\sum_{j=1}^{J} I_{Y_{ij} \neq NA} \cdot q_{jk}} \qquad \text{where } i = 1, 2, ...N; k = 1, 2, ..., K \tag{1}$$

where Y_{ij} and q_{jk} are the corresponding entries from the response matrix Y and Q-matrix.

There are several benefits of using a summary statistic of this form. The statistic scales for the number of items in which the skill appears as well as for missing data. If a student has not seen all or any of the items requiring a particular skill, we still derive an estimate based on the available information. Also, the values naturally fall onto a skill mastery scale. For each skill, zero indicates that a student has not mastered that skill, one indicates that they have, and 0.5 indicates uncertainty, partial mastery, or no information.

Moreover, the vectors $B_i = \{B_{i1}, B_{i2}, ..., B_{iK}\}$ for $i = 1, 2, ..., N$, lie in a K-dimensional unit hyper-cube where each skill corresponds to a dimension and each corner is one of the 2^K natural skill set profiles $C_i = \{C_{i1}, C_{i2}, ..., C_{iK}\}$, $C_{ik} \in \{0, 1\}$. In Figure 1, we show the corresponding hyper-cubes for $K = 2, 3$ with selected profile locations labeled. For example, if $K = 3$ and a student has the first two skills but not the third, their true skill set profile would be $\{1, 1, 0\}$, the triangle in the bottom right back corner in Figure 1(b). The B_i's map each student into the unit hyper-cube. Ideally, students would be represented with a point mass at each of the 2^K corners. However in practice, students will not map directly to the corners due to error, they may guess without having the skills or they may have the skills and slip. In the capability matrix and the corresponding hyper-cube, values near or at zero and one indicate certainty about skill mastery (no/yes). We are less certain about skill mastery for values near 0.5. Note that multiple students may map to the same locations.



(a) 2D hyper-cube with 4 true skill set profiles

(b) 3D hypercube with 8 true skill set profiles

(c) B-matrix example

Figure 1: Examples of 2D and 3D hyper-cubes with skill set profiles and a 2D B-matrix.

Suppose that we have the response matrix Y, the Q-matrix, and corresponding B-matrix shown below. Figure 1(c) illustrates the corresponding mapping of the three students into the two-dimensional hyper-cube. For student 1 at $\{0.8, 0.2\}$, we might say they likely have skill 1 but likely do not have skill 2. For student 2 at $\{0.2, 0.33\}$ we might say they likely do not have skill 1 or skill 2, but we are less certain about their skill 2 status. Finally, for student 3 at $\{0.83, 1\}$, we would say that they likely have skill 1 and definitely have skill 2.

$$Y = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & NA & 1 \\ 0 & 0 & NA & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad B = \begin{pmatrix} B_{11} = \frac{4}{5} & B_{12} = \frac{1}{5} \\ B_{21} = \frac{1}{5} & B_{22} = \frac{2}{6} = \frac{1}{3} \\ B_{31} = \frac{5}{6} & B_{32} = \frac{6}{6} = 1 \end{pmatrix}$$

$$Q^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

3 Clustering Methods

To identify groups of students with similar skill set profiles, we cluster the rows of the B matrix (and subsequently partition the hyper-cube) using two commonly used clustering procedures: k-Means and model-based clustering. While other methods are available, characteristics of these clustering procedures make them natural choices for this problem.

3.1 k-Means (With Empty Clusters)

k-Means [4] is a popular iterative descent algorithm for data $X = \{\underline{x}_1, \underline{x}_2 ..., \underline{x}_n\}$, $\underline{x}_i \in \Re^K$. It uses squared Euclidean distance as a dissimilarity measure and tries to minimize within-cluster distance and maximize between-cluster distance. For a given number of clusters G, k-Means searches for cluster centers m_g and assignments A that minimize the criterion

$$\min_A \sum_{g=1}^{G} \sum_{A(i)=g} \|\underline{x}_i - m_g\|^2.$$

The algorithm alternates between optimizing the cluster centers for the current assignment (by the current cluster means) and optimizing the cluster assignment for a given set of

cluster centers (by assigning to the closest current center) until convergence (i.e. cluster assignments do not change). It tends to find compact, spherical clusters and requires *a priori* both the number of clusters G and a starting set of cluster centers. The final cluster assignment can be sensitive to the choice of centers; a common method for initializing k-Means is to randomly choose G observations. However, in our hyper-cube, we have a natural set of starting cluster centers, the 2^K skill set profiles at the corners. If students map closely to their profile corners, k-Means should locate the groups affiliated with the corners quickly. However, if we are missing students from one or more skill set profiles in our population, forcing $G = 2^K$ clusters will split some clusters unnecessarily. We modify the k-Means algorithm to allow for empty clusters (or absent skill set profiles) in the following way:

1. Set the starting cluster centers m_g to the corners of the K-dim hyper-cube (2^K centers).
2. Create the cluster assignment vector A by assigning each B_i to the closest m_g.
3. For all clusters g, if no B_i is assigned to m_g, i.e. $\sum I_{A(i)=g} = 0$, then m_g remains the same. Else, $m_g = \frac{\sum_{i=1}^{n} I_{A(i)=g} \cdot B_i}{\sum_{i=1}^{n} I_{A(i)=g}}$.
4. Alternate between 2) and 3) until the cluster assignment vector A does not change.

This flexible k-Means variation allows for empty clusters or fewer clusters than originally requested and removes the constraint that there be one cluster per skill set profile.

3.2 Model-based Clustering

Model-based clustering [3,9] is a parametric statistical approach that assumes: the data $X = \{\underline{x}_1, \underline{x}_2, ..., \underline{x}_n\}$, $\underline{x}_i \in \Re^K$ are an independently and identically distributed sample from some unknown population density $p(\underline{x})$; each population group g is represented by a (often Gaussian) density $p_g(\underline{x})$; and $p(\underline{x})$ is a weighted mixture of these density components, i.e.

$$p(\underline{x}) = \sum_{g=1}^{G} \pi_g \cdot p_g(\underline{x}; \theta_g)$$

where $\sum \pi_g = 1$, $0 < \pi_g \leq 1$ for $g = 1, 2, ..., G$, and $\theta_g = (\mu_g, \Sigma_g)$ for Gaussian components. The method finds estimates for the number of clusters G as well as their centers and variances (μ_g, Σ_g) that maximize a chosen information criterion. Essentially, it finds the weighted combination of Gaussian densities that "best fits" the data. While it may require the groups to have Gaussian densities, it is very flexible (unlike k-Means) on the shape, volume, and orientation of the densities. This freedom allows model-based clustering to fit a wide array of student groups of different shapes and sizes.

Both methods return a set of cluster centers and variances and an assignment vector mapping each B_i to a cluster. They do not, however, automatically assign a natural skill set profile (hyper-cube corner) to each cluster. Ideally, we have 2^K clusters, each closest to a unique corner. In reality, some corners will have no students nearby. The k-Means algorithm has been altered to allow for this option; model-based clustering estimates centers in high-frequency areas and should not put a center near an empty corner. We do not advocate a one-to-one mapping of clusters to corners; clusters near areas of uncertainty in the hyper-cube should be identified as such. If a cluster of students is centered at $\{0.12, 0.88, 0.55\}$,

they should be labeled as likely not having skill 1, likely having skill 2, and uncertain on skill 3. This conservative classification will help teachers avoid misclassifying students. To classify a new student, we calculate the capability vector and assign to the nearest cluster.

3.3 Subspace Clustering

If few items require skill k, B_{ik} only take a few unique values. For example, if three items need skill k, $B_{ik} \in \{0, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\}$. Clustering on the K-dimensional hyper-cube may not perform well as students will map to only a few (K-1)-dimensional hyper-cubes. Instead we recommend conditioning on the coarsely gridded dimension (skill k, where students are already well-separated) and clustering on the (K-1)-dimensional conditional subspaces (repeating as needed).

4 Examples

For our simulated data, we use the deterministic inputs, noisy "and" gate model (DINA; [8]) a conjunctive cognitive diagnosis model. The DINA model item response form is

$$P(Y_{ij} = 1 \mid \eta_{ij}, s_j, g_j) = (1 - s_j)^{\eta_{ij}} g_j^{1-\eta_{ij}}$$

where $\alpha_{ik} = I_{\{\text{Student } i \text{ has skill } k\}}$ indicates if student i possesses skill k, $\eta_{ij} = \prod_{k=1}^{K} \alpha_{ik}^{q_{jk}}$ indicates if student i has all skills needed for item j, for item j, $s_j = P(Y_{ij} = 0 \mid \eta_{ij} = 1)$ is the slip parameter and $g_j = P(Y_{ij} = 1 \mid \eta_{ij} = 0)$ is the guess parameter. If a student is missing any of the required skills, the probability that they will answer an item correctly drops due to the conjunctive assumption.

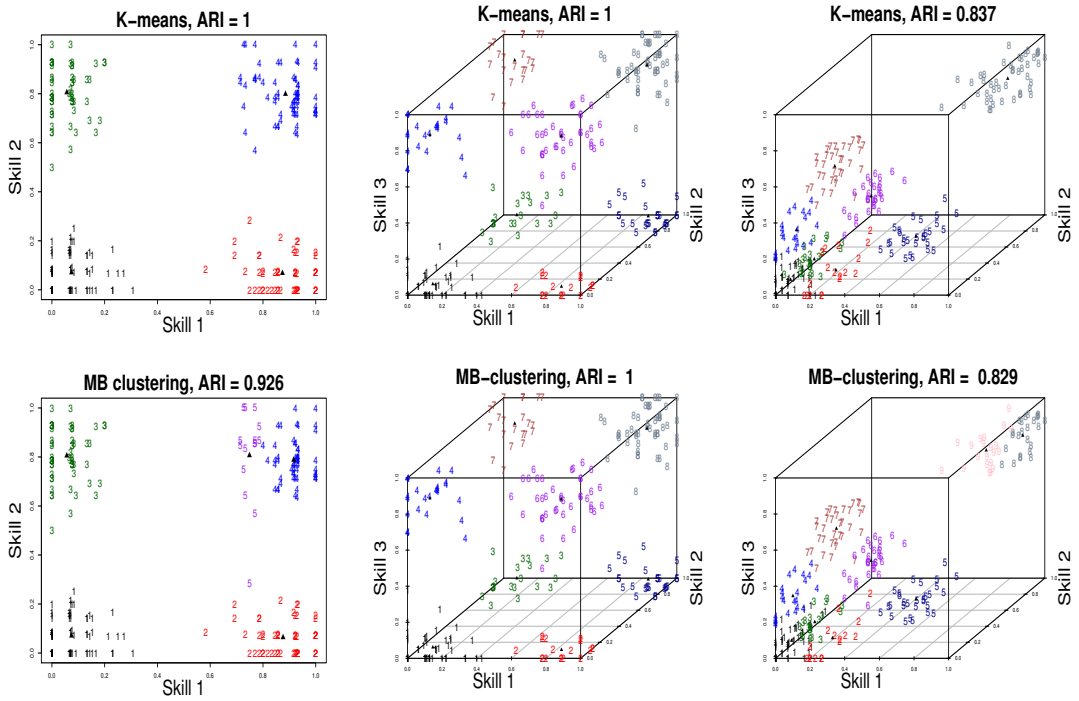
When simulating data from the DINA model, we first fix skill difficulties and inter-skill correlation and generate true skill set profiles C_i for each student. If skills are of equal difficulty with little or no inter-skill correlation, students are evenly spread among the 2^K natural skill set profiles. If skill difficulty varies, skill set profiles with only "easy" skills will have more students than those including the "hard" skills. High inter-skill correlation pulls students toward the no mastered skills and all mastered skills corners ($C_i = \{\underline{0}\}, \{\underline{1}\}$). Next we draw slip and guess parameters from a random uniform distribution ($s_j \sim$ Unif(0,0.30); $g_j \sim$ Unif(0,0.15)). Given profiles and slip/guess parameters, we generate the student response matrix Y. Prior to clustering, we remove 10% of the responses completely at random.

For these examples we know the true underlying skill set profiles C_i and can calculate their agreement with the clustering partitions using the Adjusted Rand Index (ARI; [7]), a common measure of agreement between two partitions. The expected value of the ARI is zero and the maximum value is one, with larger values indicating better agreement.

4.1 Simulated DINA Data

In Example 1, we generated response data for $N = 250$ students for $J = 30$ items, $K = 2$ skills. The Q-matrix contains only single skill items, 15 items per skill. The skills are equal difficulty with an inter-skill correlation of 0.25. Figure 2(a) shows the results. Clusters are number/color coded with triangle centers. We asked k-Means for $2^K = 4$ clusters; all students were clustered correctly (ARI = 1). Model-based clustering chooses five clusters

(ARI = 0.926). The "extra" high frequency area near $\{1, 1\}$ results from the close proximity or identical locations of the 19 students in Cluster 5. Teachers could interpret these results as two groups with similar skill 2 mastery but different skill 1 mastery.



(a) Ex 1: 2 skills; all single skill items

(b) Ex 2: 3 skills; all single skill items

(c) Ex 3: 3 skills; single, double and triple skill items

Figure 2: Simulated data examples for $K = 2, 3$ skills, single skill and multiple skill items, 10% missing responses. Clusters are color/number coded, centers denoted by triangles.

In Example 2, we simulated as in Example 1 but increased the number of skills to $K = 3$. Again the Q-matrix was designed to only include single skill items, 10 items per skill. Here, both k-Means and model-based clustering recovered the true skill set profiles (ARI=1). Figure 2(b) shows the clustering results for both methods.

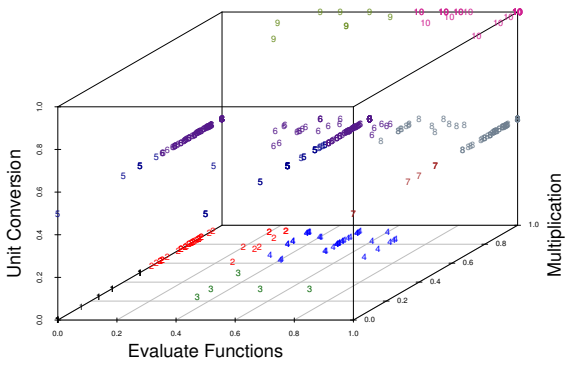
For Example 3, we simulated as in Example 2 but used a balanced design Q-matrix including multiple skill items where each skill appeared by itself in four items, in four double skill items with each of the other two skills, and in three triple skill items. Results are in Figure 2(c). Both methods find clusters of students showing mastery of all three skills in the back upper right corner near the $\{1, 1, 1\}$ skill set profile. However, the remaining students are pulled toward the front lower left corner (the $\{0, 0, 0\}$ skill set profile), a direct result of the combination skill items. If a student incorrectly answers a multiple skill item, all skills required by that item are penalized (not just the unmastered skills). We have seen that a balanced design negates the penalty effect (ARI = 0.837, 0.829); the remaining clusters are effectively scaled and maintain their separation.

The datasets presented are missing 10% of the responses; we compare their results to those for only students not missing any responses. In educational data mining, we commonly use case-wise deletion of students to generate a complete dataset. This method is

impractical here as it leaves us with 11, 10, and 15 students respectively. Instead we use the original generated response matrices prior to removing responses at random. The B-matrices are re-calculated and clustered. Only Example 3 had different ARIs. When using the complete data set, the ARI for k-Means increases from 0.837 to 0.880, for model-based clustering, 0.829 to 0.946. These jumps are expected as the lack of missingness increases the number of items seen (and the fineness of the grid) and decreases the relative effect of the penalty associated with incorrectly answering a multiple skill item; the resulting clusters are less removed from the corners.

A higher dimensional example with $N = 1000$ students, $J = 80$ items, and $K = 20$ skills was also explored. In this case there were 425 unique latent classes used to generate the data. Model-based clustering found 424 clusters and had an ARI of 0.99. Giving k-means 2^{20} starting centers is unreasonable; we're currently developing methods to systematically and appropriately choose a smaller set of starting centers.

4.2 Assistment Data



Cluster / Skill	Eval Fct	Mult	Unit Conv
1	0	0.23	0
2	0.05	0.79	0
3	0.43	0.35	0
4	0.47	0.83	0
5	0.40	0.49	0.5
6	0.29	0.92	0.5
7	1.0	0.32	0.5
8	0.94	0.90	0.5
9	0.43	0.91	1
10	0.91	0.98	1

Figure 3: Assistment System example of conditional k-Means clustering on the B-matrix; clusters are color/number coded. The table shows the cluster centers.

For our real data, we use a subset of 26 items requiring three skills (for easy visualization) from the Assistment System online mathematics tutor [5]. The Q-matrix is unbalanced; Skill 1 (Evaluating Functions) appears in eight items, Skill 2 (Multiplication) in 20 items, and Skill 3 (Unit Conversion) in two items. Overall, 551 students answered at least one item, however there is a large amount of missing data (57%). Recall, if student i did not see any items requiring skill k, $B_{ik} = 0.5$. Since Unit Conversion appears in only two items, $B_{iUC} \in \{0, \frac{1}{2}, 1\}$. The three corresponding planes are visible in Figure 3. We condition the unique B_{iUC} values and apply our k-Means variation (Section 3.1) to each plane. The final cluster centers are in the table in Figure 3. k-Means is preferable here because the limited number of unique values in the Evaluate Functions skill dimension leads to instability in the more flexible model-based clustering models. The planes corresponding to $B_{iUC} = 0$ and 0.5 each have four clusters; the plane for $B_{iUC} = 1$ has two. There are natural interpretations for each of the clusters. For example, a teacher might interpret Cluster 9 as students who know Unit Conversion and Multiplication, but are uncertain on Evaluating Functions. Cluster 10 could be interpreted as the students who have mastered all three skills.

5 Conclusions and Future Work

We derived a capability matrix to summarize student skill mastery for use in clustering algorithms. In simulated datasets, the method performed well (i.e., high values of ARI). In the Assistments data the method responded well to missing data, allowing us to draw conclusions for the skills that students have seen and distinguish the skills that require more assessment. Early results suggest that the Q-matrix design plays a large role in the location and interpretation of the clusters. Finally, we visually presented examples with $K = 2$ and $K = 3$ skills and showed the method scales to a larger number of skills.

Currently, we are comparing our results to other student skill knowledge estimates. For example, using WinBUGS [11], the DINA model estimates produce essentially the same profile clusters for the simulated datasets; however, it runs around 700 times more slowly.

References

[1] Anozie, N.O. and Junker, B. W. (2007). *Investigating the utility of a conjunctive model in Q-matrix assessment using monthly student records in an online tutoring system.* National Council on Measurement in Education (NCME-07), April 12, 2007, Chicago, IL.

[2] Barnes, T.M. (2003). *The Q-matrix Method of Fault-tolerant Teaching in Knowledge Assessment and Data Mining.* Ph.D. Dissertation, Department of Computer Science, North Carolina State University.

[3] Fraley, C. and Raftery, A. (1998). How many clusters? which clustering method? - answers via model-based cluster analysis. *The Computer Journal*, 41, 578-588.

[4] Hartigan, J. and Wong, M.A. (1979). A k-means clustering algorithm. *Applied Statistics*, 28, 100-108.

[5] Heffernan, N.T., Koedinger, K.R. and Junker, B.W. (2001). *Using Web-Based Cognitive Assessment Systems for Predicting Student Performance on State Exams.* Research proposal to the Institute of Educational Statistics, US Department of Education. Department of Computer Science at Worcester Polytechnic Institute, Worcester County, Massachusetts.

[6] Henson, J., Templin, R., and Douglas, J. (2007). Using efficient model based sum-scores for conducting skill diagnoses. *Journal of Education Measurement*, 44, 361-376.

[7] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2, 193-218.

[8] Junker, B.W. and Sijtsma K. (2001). Cognitive Assessment Models with Few Assumptions and Connections with Nonparametric Item Response Theory. *Applied Psych Measurement*, 25, 258-272.

[9] McLachlan, G.J. and Basford, K.E. (1988). *Mixture Models: Inference and Applications to Clustering*.

[10] Nichols, P.D., Chipman, S.F., and Brennan, R.L. (1995). *Cognitively Diagnostic Assessment.* Lawrence Erlbaum Associates.

[11] Spiegelhalter, D.J., Thomas, A. and Best, N.G. (2003). *WinBUGS: Bayesian Inference Using Gibbs Sampling, Manual Version 1.4*. Cambridge: Medical Research Council Biostatistics Unit.

[12] Talavera, L., and Gaudioso, E. (2004). Mining student data to characterize similar behaviour groups in unstructured collaboration spaces. *Proceedings of the Artificial Intelligence in Computer Supported Collaborative Learning Workshop at the ECAI 2004.* Valencia, Spain.

[13] Tatsuoka, K.K. (1983). Rule Space: An Approach for Dealing with Misconceptions Based on Item Response Theory. *Journal of Educational Measurement*. Vol. 20, No. 4, 345-354.

Can we Predict which Groups of Questions Students will Learn from?

Mingyu Feng[1], Neil Heffernan[1], Joseph E. Beck[1], Kenneth Koedinger[2]

[1]{mfeng, nth}@wpi.edu, joseph.beck@educationaldatamining.org
Computer Science Department, Worcester Polytechnic Institute
[2]koedinger@cmu.edu
Human Computer Interaction Institute, Carnegie Mellon University

Abstract. In a previous study ([4]), we used the ASSISTment system to track student knowledge longitudinally over the course of a schools year, based upon each student using our system about a dozen times during the course of the year. This result confounded learning from the computer system with students learning from their sitting their normal class. In this work, we look to see if students were reliably learning from their time spent with the *computer in a single day*. Our result suggests that students performed better later in the same computer session on similar skills, which indicates students are learning from using ASSISTments. However, learning is rather uneven across groups of skills. We test a variety of hypotheses to explain this phenomenon and found that the automated approaches we tried were unable to account for the variation. However, human expert judgments were predictive as to which groups of skills were learnable.

1 Introduction

The field of educational data mining is often concerned with how to model student learning over time. More often than not, these models are concerned with how student performance changes while students are using the computer. In this project we look to see if learning from the computer system was happening over time, trying to separate out learning from the classroom. We then wanted to investigate to see if we could predict on which knowledge components students were systematically learning.

The ASSISTment project was funded to see if it was possible to teach students effectively while assessing student performance accurately at the same time. We have reported the results of our analysis of the assessment value of our system in [4] . The results indicated that student performance is on average reliably increasing during the course of the year. But since most students only use the system every other week, it is unclear how much credit should go to ASSISTments vs. classroom instruction.

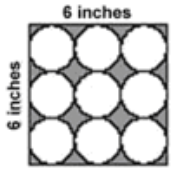
In order to track students' progress in learning, different approaches have been established. Corbett & Anderson [3] employed a diagnostic approach called *knowledge tracing* that models students as an overlay of the ideal production rules. They proposed a two-state (learned & unlearned) learning model that allows student knowledge state to transit from one to the other probabilistically. One technique by Koedinger and colleagues is called Learning Factors Analysis (LFA) [2] takes advantage of the Power Law of Learning (see [7]) to fit student performance to a power function reflecting decreasing error rates over time. LFA has been proposed as a generic solution to evaluate

and compare many potential cognitive models of learning. Since student performance was often represented by a dichotomous variable, logistic regression models have been used as the statistical model for evaluation (e.g. [2] , [6]). In terms of related work on investigating the reasons of learning, Vanlehn et al. [8] explored the problem of what causes learning by contrasting cases where tutoring does or does not result in learning. In this study, we will investigate whether students learn within ASSISTments. We conducted a focused analysis of a subset of items and tracked how student performance on these items changes during the same ASSISTment session. We will explore the possible reasons of why on some sets of problems students learned or failed to learn.

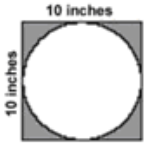
2 Methodology

2.1 Experimental Design

Our hypothesis was that students were learning groups of items that share the same background knowledge requirement. Our subject manner expert picked 182 items out of the 300 8th grade (approximately 13 to 14 years old) math items in ASSISTment. Items that have the same deep features or knowledge requirements, like approximating square roots, but have different surface features, like cover stories, were organized into a **Group of Learning OPportunity (GLOP).** Besides, the expert excluded groups of items where learning would be too obvious or too trivial to be impressive. Singleton items were not selected either. The selected 182 items fall into 40 GLOPs with the number of items in each GLOP varies from 2 to 11. The items cover knowledge from all of the five major content strands identified by the Massachusetts Mathematics Curriculum Framework, relatively heavy on the strand Patterns, Relations & Algebra. Items in the same group were collected into the same section of ASSISTments, and seen in random order by students. Each student potentially saw 40 different GLOPs that involve different 8th grade math skills (e.g. fraction-multiplication, inducing-functions, symbolization articulation) in random order. Figure 1 shows three items in one GLOP that are about the concept "Area." All these problems asked students to compute the area of the shaded part in the figures. It is worth pointing out that all the GLOPs were constructed by focusing on the content of the items before the analysis done in this paper.



What is the area of the shaded part of this figure? Assume π = 3.14.



What is the area of the shaded part of this figure? Assume π = 3.14.

What is the area of the shaded region in the figure above? (Use 3.14 for pi.)

Figure 1. A sample GLOP that addresses the skill "Area"

We assessed learning by comparing student performance the first time they were given one item from a GLOP with their performance when they were given more items (also more opportunities) from the same GLOP in the same day. If students tend to perform

better on later opportunities of items in a GLOP, it indicates that they may have learned from the instructional assistance provided on items by the ASSISTment system that they worked on earlier by answering the scaffolding questions or by reading hint messages. There is controversy over whether same-day learning opportunities should be used as evidence of learning. For example, Beck [1] thought repeated trials were not indicative of learning. He chose not to use later encounters on the same day in the Reading Tutor since performance on those encounters is not a reflection of student knowledge but just retrieved from short term memory. Our domain (mostly 8th grade multi-step math problems) is more complex than reading and the items in a GLOP usually have different surface features. Solving these problems is not simple retrieval of an answer from a previous question. And even if it wasn't more complex, our later day trials are horribly confounded by classroom instruction due to low density of usage (every other week). In this paper, we chose to analyze the response data on the same day to eliminate the confound of learning happening because of classroom instruction **between** two ASSISTment sessions.

2.2 Sample of the data we used for analysis

We collected data for this analysis from Oct. 31, 2006 to Oct. 11th, 2007. 2000+ 8th grade students participated in the study. We defined participation in a GLOP as answering two or more questions in it, and excluded students who participated in less than five GLOPs to make sure each student has at least 10 data points. We ended up with a data set of 42,086 rows, with each row representing a student's attempt at an item. 777 students entered into our final data set, and each student on average worked on 54 items across 14 GLOPs. Table 1 shows a small sample of our data. In particular, Table 1 shows two students' performance on two GLOPs, 16, and 3 (partly). We use the column "correct?" to indicate whether the student answered the question correctly or not. The value will be 1 where he succeeded; otherwise, it is set to be zero. The first student worked on three problems (i.e. had 3 opportunities to learn) from GLOP 16 on April 2nd, 2007 starting from 9:39AM. He failed the first two but managed to solve the last one.

Table 1. Sample data showing two students' performance on two GLOPs

Student ID	GLOP ID	Question ID	Date	Time	Correct?	Opportunity
30296	16	1069	4/2/2007	9:39:20	0	1
30296	16	231	4/2/2007	9:42:19	0	2
30296	16	1512	4/2/2007	9:53:11	1	3
30300	3	2267	4/25/2007	7:48:15	0	1
30300	3	2244	4/25/2007	7:58:47	1	2

3 Research Question 1: Do students learn from ASSISTments?

We first attempted to determine whether the system effectively teaches. To answer the first research question if students are learning from ASSISTments, we ran a logistic regression to study the relationship between student performance (i.e. their responses to items) and the number of opportunities the student has on a GLOP. In our method, the dependent variable is student response to a question and we account for the difference of

student math proficiency by including the student as one of the predictor variables. Similarly, we include the question as another predictor with regard to the fact that questions in one GLOP may vary in difficulties. The regression formula is

Equation 1. Logistic regression model

$$\ln(\frac{p_{ij}}{1-p_{ij}}) = \alpha_i * Student_i + \beta_j * Question_j + \gamma * Opportunity\#$$

Where p_{ij} is the probability that the student *i* will answer question *j* correctly

Opportunity# indicates how many opportunities the student *i* has on a particular GLOP.

α_i, β_j and γ are the coefficients for the corresponding predictors $Student_i$, $Question_j$ and *Opportunity#.*

The model is very similar to LFA models except that skills are not included as factors since we are investigating generalized learning over all GLOPS. We ran a multinomial logistic regression treating student and question as factors and opportunity as a covariate. The regression coefficient estimated by the model, corresponding to the number of learning opportunities (γ), is .03 ($p < .001$). This result suggests that in general, students performed reliably better as they have more chances of practicing on the same GLOP. This result suggests that in general, students performed reliably better as they have more chances of practicing on the same GLOP. The coefficient in the logistic regression model indicates that students will improve by 0.03, on a logit scale, for each practice opportunity. This learning corresponds to approximately a 0.8% improvement in performance for each problem practiced, a rather small effect. In Massachusetts, MCAS test scores are categorized into four performance levels (namely warning, need improvement, proficient, and advanced). According to the results of 2006 MCAS test, students need to earn 13 more points (24% of the full score) to jump from need improvement to proficient which is required by the federal movement based on NCLB standards to graduate from high school. Theoretically, if students can gain 0.7% for each learning opportunity, they will fulfill the 24% improvement by solving 31 problems in ASSISTments. It should be noted that there may be a selection effect in this experiment in that better students are more likely to do more problems in a day and therefore more likely to contribute to this analysis. Also there is a limitation with the model that all GLOPs are assumed to produce the same amount of learning, which may not be true as we will show later.

A positive answer to the first question allows us to claim that students are learning from working in ASSISTments and the learning results are generalized across the 40 GLOPs. Then, we stepped further to explore if all of the GLOPs are equally effective at promoting learning. The answer is “no”, which is not surprising anyway since the items in different GLOPs vary on several aspects (e.g. focusing on various skills; built by authors with differing teaching experience using various teaching strategies, etc.). In summary, out of the 40 GLOPs, the amount of learning per opportunity is statistically reliably higher than zero on 11 of them. 2 GLOPs caused marginally reliable learning and 16 caused unreliable learning. And there is non-reliable “un-learning” for the remaining 11 GLOPs, suggesting that not much learning occurred when students worked on these GLOPs.

4 Research Question 2: Why students learned or failed to learn?

Now that we have shown that learning varies among GLOPs, we will explore the reasons for this variation. We are not only interested to know which category each GLOP falls in and but also curious why. Particularly, we want to investigate why students did not show learning on certain GLOPs. Our four hypotheses are:

1. H1: Learning transfer from harder items to easier items, or students tend to learn more by doing harder items than by doing easier items. Presumably, if a student learned to solve a hard item, he then should be able to do better on an easier item that requires similar skills. However, the converse is not necessarily true.
2. H2: Knowledge transfer occurs within GLOPs of items that use similar skills. We can never know exactly how a student internally represents a problem and what the exact skills a student applied to solve a problem. But if a GLOP is well-focused in what it covers, presumably students should show more learning within it.
3. H3: The "learnability" of the skills required by GLOPs varies. Our statistics show that each ASSISTment provides about 2 minutes of instruction. It can be hard to teach some skills effectively, for instance, *symbolization articulation*, in such a short period. Such skills require deep understanding and more practice to be able to apply and transfer, whereas some other skills such as *area* are more teachable since students only need to be reminded to apply the area formula.
4. H4: The efficacy of instructions has an impact on learning results. We can easily imagine that some GLOPs have better teaching efficacy than others. The quality of the scaffolding questions and hint messages can differ from one item to another as authors used a tutoring strategy that are more, or less, effective than others.

In this paper, we will test the first two hypotheses and leave the last two as future work. We plan to invite more content experts to help us identify the learnability of the related skills and to evaluate the quality of the ASSISTments by looking closely at the scaffolding questions and hint messages.

4.1 Do students learn more from harder items or easier items?

Noticing learning varies among GLOPs, the first thing we did is to explore the relationship between the amount of learning and the easiness of a GLOP (measured by the average difficulty of items in the GLOP). We calculated the rank-order correlation and got a coefficient of .333 ($p = .036$, $N=40$), which indicates that students learned more on harder GLOPs than on easier ones. We asked ourselves: why is this? A quick answer is that there is more room to grow for harder items. Or, maybe students just learn more from harder items than easier items.

Beck [1] introduced an approach called *learning decomposition* to analyze what type of practice was most effective for helping students learn a skill. The approach is a generalization of learning curve analysis, and uses regression to determine how to weight different types of practice opportunities relative to each other. In this paper, we apply learning decomposition to our data set to investigate how students acquire math skills: will their practice on harder items produce more learning? To test our first hypothesis, we added two columns to our data set. One column, entitled "easier_before_current",

represents how many items the student has seen in the same GLOP are easier than the current item. The other column, entitled "harder_before_current", indicates how many items were seen that are harder than the current one. We measure the easiness of the items using the item parameter given by a one-parameter Item Response Theory model (i.e. Rasch model[1]). The Rasch model was trained over data collected in the system from Sept., 2004 to Jan., 2008, including responses to 2,700 items from more than 14,000 students. We include the two columns as covariates in the regression model.

Equation 2. Learning Decomposition Model

$$\ln(\frac{p_{ij}}{1-p_{ij}}) = \alpha_i * Student_i + \beta_j * Question_j + \gamma_e * easier\_before\_current + \gamma_h * harder\_before\_current$$

Where γ_e and γ_h represents the coefficients for the two new covariates respectively.

The model was fitted in SPSS 14.0. We noticed that the coefficients of the two covariates of *easier_before_current* and *harder_before_current* are very close to each other (γ_e = .032, and γ_h = .033). The coefficient for *easier_before_current* is fractionally but not reliably lower (p=.966), which suggested that students learn as much from easier items as from harder items, and thus our first hypothesis is rejected.

4.2 Does more learning occur in GLOPs that are more focused?

H2 is different than H1 in that it believes that transfer occurs within GLOPs that have similar difficulty questions (and therefore address similar skills based on our assumption). To test H2, we want to investigate the relationship between the amount of learning that happened in each GLOP and the cohesiveness of the GLOP in term of item difficulty and the skills that are needed to answer the items.

We used two approaches to quantify the cohesiveness of the GLOPs. The first metric is an automated measure that comes from a computer modeling process based on the assumption that if two skills, A and B, are better modeled by a single skill, then practice on either A or B symmetrically transfers to the other. During the modeling process, for each GLOP, we compared the BIC of two models. The first model treated each question as having a separate difficulty. The second model treated all questions as having the same difficulty, and thus had *number_of_questions_in_GLOP-1* fewer parameters. Presumably, if the cohesiveness of a GLOP is high, we should expect the second model to fit better on our data as measured by Bayesian Information Criteria (BIC) (or any model fitting criteria that penalizes for model complexity). We followed the same procedure and calculated the difference of BIC values between two models for each of the GLOPs.

The second metric is based on our subject matter expert's ranking of the cohesiveness of the GLOPs. As requested by us, our subject matter expert set the rating from 1 to 5. A fit

[1] In the Rasch model, the probability of a specified response is modeled as a logistic function of the difference between the person and item parameter. In educational tests, item parameters pertain to the difficulty of items while person parameters pertain to the ability or attainment level of people who are assessed.

of 1 or 2 means that the items are very different. A 3 means there are some flaws in the selection, a 4 means there are just a few inconsistencies and a 5 means they fit very well. According to the ranking of the expert, 18 GLOPs got a fit of 5, 10 were given a fit of 4, 7 GLOPs got 3 and the remaining 5 GLOPs scored 2.

After obtaining the two metrics, we continued to analyze the relationship between the cohesiveness of the GLOPs and the amount of learning that happened in each of them. First, we calculated the rank-order correlation between the automated metric and the amount of learning (given by Equation II) but did not find a significant relation ($r = .13$, $p = .94$). We then discretized coherence into 3 coherence bins: high, medium and low and performed a one-way ANOVA to explore whether there were any differences in the amount of learning, but found no main effect ($F = .676$, $p = .515$). After that, we did the same analysis using the expert ranking of the cohesiveness. The rank-order correlation between fit and amount of learning is equal to .322 ($p = .045$). Yet the ANOVA shows no main effect of fit ($F = 1.573$, $p = .213$). Further more, instead of using five groups, we merged all GLOPs with fit less than 5 into one category named "non-perfect-fit" as a contrast to the ones with "perfect-fit" and ran an independent sample t-test to compare the mean between the two categories. The result suggested that there is statistically reliably more learning happening in GLOPs of perfect fit ($t = 2.311$, $p = .030$).

To complete the third side of the triangle of learning/automated coherence metric/expert ranking, we also computed the correlation between our two metrics of fit/cohesiveness and found out that they do not correlate with each other ($r =-.198$, $p=.22$), which means that an automated measure and an expert's judgment differ. In conclusion, H2 was supported by the expert's judgment but not by the result of data mining.

5 Future work and Conclusions

In terms of caveats and recognized limitations, we want to first acknowledge that we don't have control group to compare the learning result against to. Also, if student performance systematically varies over time apart from learning, our model is not able to account for it. For instance, if students experienced a ramp up effect of doing better over time, this could explain away our results. Similarly, if students get fatigued over a class period we would be underestimate the learning effect.

As a future work, we want to look to see how the items would be grouped by some automated method such as Q-matrix algorithm or LFA.

In conclusion, we presented evidence that suggests there is learning within ASSISTments. More interestingly, we found that the learning differed across the groups of items. We tested a variety of hypotheses to explain this phenomenon and found that the automated approaches we tried were unable to account for the variation. However, human expert judgments were predictive as to which groups of skills were learnable.

The contribution of the paper lies in two aspects. First, we looked at when learning occurs in an intelligent tutoring system and examined a variety of hypotheses on why learning happens. While these hypotheses seemed intuitive, they were not supported by our analysis. Second, student modeling research typically accounts for the amount of learning due to a practice opportunity, but generally does not try to take into account of

learning outside the tutor (such as classroom instruction, homework, etc.). Using this type of analysis that focuses on within-session learning, we isolated the effect to those caused by our tutor.

Acknowledgement

This research was made possible by the U.S. Department of Education, Institute of Education Science (IES) grants, "Effective Mathematics Education Research" program grant #R305K03140 and "Making Longitudinal Web-based Assessments Give Cognitively Diagnostic Reports to Teachers, Parents, & Students while Employing Mastery learning" program grant #R305A070440, the Office of Naval Research grant # N00014-03-1-0221, NSF CAREER award to Neil Heffernan, and the Spencer Foundation. All the opinions, findings, and conclusions expressed in this article are those of the authors, and do not reflect the views of any of the funders.

References

[1] Beck, J.E. (2006). Using learning decomposition to analyze student fluency development. *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems*. Jhongli, Taiwan. Pages 21-28.

[2] Cen, H., Koedinger, K., & Junker, B. (2006). Learning factor analysis – A general method for cognitive model evaluation and improvement. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. Springer-Verlag: Berlin. pp. 164-175.

[3] Corbett, A. T. & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4, 253-278.

[4] Feng, M., Heffernan, N.T, Koedinger, K.R. (2006). Addressing the testing challenge with a web-based e-assessment system that tutors as it assesses. In *Proceedings of the 15th International World Wide Web Conference*. pp. 307-316. ACM Press: New York, NY. 2006.

[5] Koedinger, K. R., Anderson, J. R., Hadley, W. H. & Mark, M. A. (1995) Intelligent tutoring goes to school in the big city. In *Proceedings of the 7th Conference on Artificial Intelligence in Education*, pp. 421-428. Charlottesville, VA: Association for the Advancement of Computing in Education.

[6] Leszczenski, J. M. & Beck J. E. (2007). What's in a Word? Extending Learning Factors Analysis to Model Reading Transfer. In *Proceedings of the Educational Data Mining workshop held at the 14th International Conference on Artificial Intelligence in Education.*

[7] Newell, A. & Simon, H.A. (1972). *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall.

[8] VanLehn, K., Siler, S., Murray, C., & Baggett, W. (1998). What makes a tutorial event effective? In M. A. Gernsbacher & S. Derry (Eds.), *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society* (pp. 1084-1089). Hillsdale, NJ: Erlbaum

Developing a Log-based Motivation Measuring Tool

Arnon Hershkovitz and Rafi Nachmias[1]
{arnonher, nachmias}@post.tau.ac.il
[1] Knowledge Technology Lab, School of Education, Tel Aviv University, Israel

Abstract. The purpose of this study is to develop a conceptual framework and a tool for measuring motivation of online learners. The study was carried out in three phases, the first of which was the construction of a framework, based on an extensive literature review. Phase two consisted of identifying variables computable from log file data, associate with the framework and compatible with previous empirical research. For this purpose, an empirical study was designed and a specific learning environment focusing on vocabulary for adults was chosen. Log files of a large population (N=2,162) were collected and variables were identified using *Learnograms*, a visual representation of learning variables over time. This phase resulted in seven explicitly defined variables, along with a mechanism to calculate them from the raw log files. The third phase included preprocessing of the dataset (reducing it to 674 cases) and application of hierarchical clustering of the variables. This phase resulted in three clusters interpreted to fit the three dimensions of motivation defined in the framework. A discussion of this study and further research is provided.

1 Introduction

Assessment of learners' motivation in online environments has been a challenge for both researchers and instructors, and the reason for it is twofold: motivation is an important factor affecting the learning process and explaining individual differences, however it is a factor difficult to evaluate without direct contact with the learner. This gap may be bridged with log file analysis, which makes it possible to learn about the online learner by means of automatically and continuously collected digital traces. Log file analysis in education research is an emerging field, however, only little research was done regarding motivation appraisal using this method.

The purpose of this study is to construct a conceptual framework and to develop a tool for measuring motivation of online learners. The motivation is built upon three dimensions to be measured based on data solely from log files: Engagement, Energization and Source. Learning variables, which describe patterns of learners' behavior, will be identified using the technique of analyzing *Learnogram*, a visual representation of learning variables over time [13]. These variables should be computable from the log files, associate with the framework, and compatible with previous empirical research. For clustering the variables, in order to classify them to the motivation dimensions, an empirical study was planned and conducted, and will be reported in this article. Thus, the main research question being addressed is: how can motivation be measured using only data from log files?

Continuing previous work done in this field, the main contributions of this study is in constructing an empirical-based automated tool for motivation detection from log files. A second contribution is the procedure in itself, since it might be transferred for developing measuring tools for other learner characteristics (e.g., anxiety, self-regulation).

2 Background

Motivation has been suggested as a factor explaining individual differences in intensity and direction of behavior [10]. It is generally accepted that motivation is "an internal state or condition that serves to activate or energize behavior and give it direction" [11]. The sources of motivation can be either internal (e.g., interestingness, enjoyment) or external (e.g., wishing for high grades, fear of parental sanctions) to the person [6].

Motivational patterns, in addition to ability, may influence the way people learn: whether they seek or avoid challenges, persist or withdraw upon difficulties, or whether they use and develop their skills effectively [7]. Different motivational patterns relate to different aspects of the learning process, e.g., achievement goals (performance or mastery), time spent on tasks, performance [1, 8, 12, 16].

Unlike configurations in which the instructor sees the students and might infer their motivation level from facial expressions, tonality - online learning supposedly disables motivation assessment. However, previous research has suggested several methods for tackling this challenge (see [4]). This research is focused on motivation measuring using information available only through log files. Table 1 summarizes the motivation-related terms and variables from five studies that mainly used learner-computer interaction data.

Following the definition given above, and based on the reviewed literature, we suggest a motivation measuring framework which considers three dimensions: a) Engagement - relates to the motivation intensity. (although using the same term, by Engagement we mean a more generalized idea than in [2, 3].); b) Energization, which refers to the way motivation is preserved and directed; c) Source of motivation (internal or external).

Table 1. Previous research on motivation recognition based on learner-computer interaction

Research	Motivation-Related Terms	Learning Variables Calculated from Logs
Beck [2]	Engagement (defined by the author)	Question response time; answer correctness
Cocea & Weibelzahl [3]	Engagement (defined by the authors)	# of pages read; time spent reading pages; # of tests/quizzes; time spent on test/quizzes
Qu & Johnson [14]	Confidence, confusion, effort	Reading time; decision time (before perform the task); task duration; # of finished tasks; # of tasks performed not from learning "plan"
de Vicente & Pain [5][1]	Control, challenge, independence, fantasy; confidence, sensory/cognitive interest, effort, satisfaction.	Quality, speed (of performance), give up
Zhang, Cheng, He, & Huang [17]	Attention, confidence	# of non-error compilations; ratio of working time and class' average; # of hints; # of executions; time until typing in editor

[1] The research was based on captured screen activity.

Within this framework, the two objectives of this study are: a) To identify variables related to the three motivation dimensions, which are computable from data stored in the system log files; b) To cluster these variables, based on empirical data, for classifying them according to the three motivation dimensions.

3 Methodology

3.1 Procedure

The study was carried out in three consecutive phases using different methodologies:

Phase I – Constructing the Research Framework. This literature-based phase was used to conceptualize the terms to be used in our motivation research. Within the framework, three dimensions were defined (Engagement, Energization, Source); see previous section.

Phase II – Identifying Variables. In order to choose and define motivation-related variables, *Learnograms* – visual representations of learning variables over time – were used as the main research tool (as described in [13]). *Learnograms* presenting students' activity (N=5) in an online vocabulary course for adults (see section 3.2) were observed, in order to identify the relevant computable variables. The compatibility of the variable to previous empirical research in this field was taken into consideration, as well as their association to our framework. At the end of this phase, seven variables were identified.

Phase III – Classifying the Variables According to the Motivation Dimensions. An empirical study for the evaluation of the identified variables was conducted (with the same learning environment used in phase I). Log files of a large population (N=2,162) for one month (April 2007) were collected and preprocessed. Students using the researched system belong to different courses (varied by length, intensity, starting date and proximity to the exam), however this logged segment was analyzed regardless the student's learning stage. A filter was applied for keeping students with at least 3 active sessions (N=1,444). Algorithms for calculating the variables were formally written and implemented using Matlab. The dataset was preprocessed and the final set of cases to be analyzed was defined (N=674). Finally, Hierarchical Clustering of the variables was applied using SPSS, with Pearson Correlation Distance as the measure and Between-groups Linkage as the clustering method.

3.2 The Learning Environment

A simple yet very intensive online learning unit was chosen as the research field. This fully-online environment focuses on Hebrew vocabulary and is accessible for students who take a face-to-face preparatory course for the Psychometric Entrance Exam (for Israeli universities). The online system is available for the participants from the beginning of the course and until the exam date (between 3 weeks and 3 months in total).

The system includes a database of around 5,000 words/phrases in Hebrew and offers varied instructional strategies: a) Memorizing, in which the student browses a table of the words/phrases along with their meanings; b) Practicing, in which the student browses the

table of the words/phrases without their meaning. The student may ask for a hint or for the explanation for each word/phrase; c) Searching for specific word/phrase; d) Gaming; e) Self testing, in the form of the exam the students will finally take. Throughout the learning process, the student may mark each word/phrase as "well known", "not-well known" or "unknown". This information is stored and being used by the system.

3.3 Log File Description

The researched system logs the students' activity, thus each student is identified by a serial number. Each row in the log file documents a session, initiating by entering the system and ending with closing the application window. For each session, the following attributes are kept: starting date, starting/ending time, number of words marked as "known" at the beginning/end of the session, ordered list of actions and their timestamps.

4 Results

4.1 Motivation-related Variables (Phase II)

The authors have examined the *Learnograms* of a few students (N=5), searching for interesting patterns and irregularities, while considering learning behavior which may be related to motivation. Seven variables were identified and calculated (see Table 2).

4.2 Classifying the Variables (Phase III)

The variable distributions were examined over the 3-sessions filtered dataset (N=1,444), see Figure 1. Three of the variables had a significant 0-values noise: *wordMarkPace*, *examPC*, *gamePC*, thus cases with 0-value in them were cleaned for focusing on the positive-value cases. Since the variables were skewed in the final dataset (N=674), we used transformations of log (*timeOnTaskPC*, *avgSession*, *wordMarkPace*, *examPC*, *gamePC*) and square-root (*avgActPace*, *avgBtwnSessions*).

Table 2. Motivation-related variables

Variable name	Variable description	Unit	Calculation remarks
timeOnTaskPC	Time on task percentage	[%]	Total time of active sessions ([min]) divided by total time frame documented.
avgSession	Average session duration	[min]	
avgActPace	Average pace of activity within sessions	[actions/ min]	Pace of activity per session is the number of actions divided by the session duration
avgBtwnSessions	Average time between sessions	[min]	
wordMarkPace	Pace of word marking	[words/ min]	Changed number of known words from beginning to end (can be negative) divided by total time frame documented
examPC	Exam activities percent	[%]	# of exam actions divided by total # of actions
gamePC	Game activities Percent	[%]	# of game actions divided by total # of actions

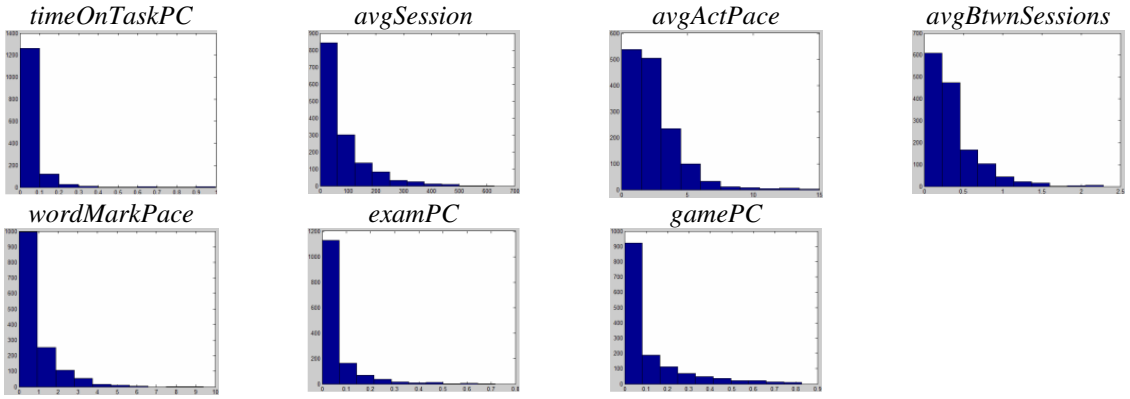


Figure 1. Distribution of the variables before cleaning and transformation were applied (N=1,444)

The clustering process is described by a dendrogram (from the Greek dendron "tree", -gramma "drawing") presented in Figure 2. The vertical lines determine which variables/clusters were grouped together and at which stage of the algorithm (from left to right). For example, the first coupled variables were *timeOnTaskPC* and *avgSession*, and next *examPC* and *gamePC* were grouped. The resulting clusters appear in Table 3, their relation to the motivation dimensions is given in the Discussion below.

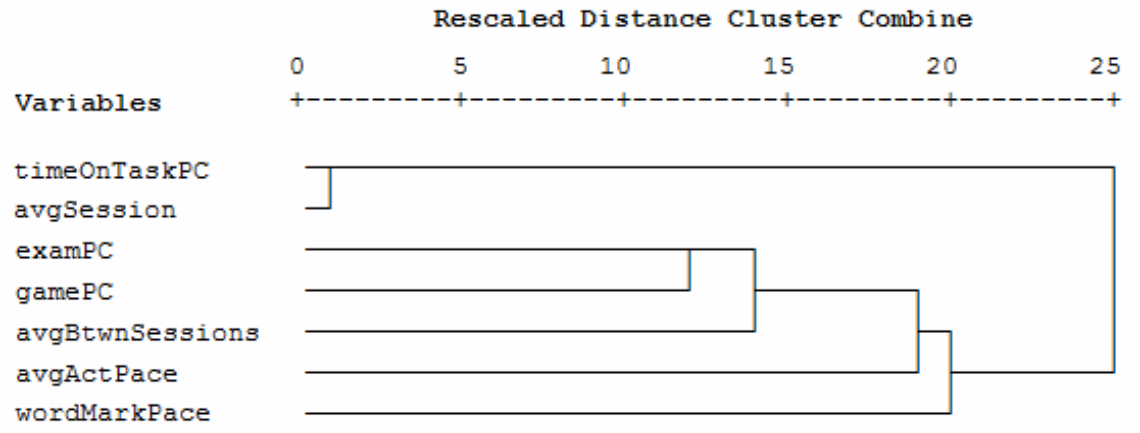


Figure 2. Dendrogram of the hierarchical clustering process

Table 3. The resulted clusters and their mapping to the motivation dimensions

Cluster	1	2	3
Variables	*timeOnTaskPC* *avgSession*	*examPC* *gamePC* *avgBtwnSessions* *avgActPace*	*wordMarkPace*
Motivation dimension	Engagement	Source	Energization

5 Discussion

In this study, an empirically-constructed tool was developed for log-based measuring of online learners' motivation. Motivation is measured by three dimensions – Engagement, Energization, and Source – and by seven computable variables corresponding to these dimensions (see Table 3). The classification of the clustered variables to the three dimensions is based on previous research in this area.

The variables *timeOnTask* and *avgSession*, which form the first cluster, might be related to the extent of Engagement, as it was previously suggested that working time might be a measure for attention or engagement [3, 17]. *examPC* and *gamePC* - grouped together in the second cluster - reflect the student's *Source* of motivation; it may be reasonable to hypothesize (inspired by, e.g., [9, 15]) that students who frequently tend to take self exams (related to performance-goal orientation) have extrinsic motivation to learn, while those who tend to game applications (related to learning-goal orientation) are intrinsically motivated. The variables *avgActPace* and *avgBtwnSessions* are also clustered together with the previous two, but their closeness to Source of motivation is yet to be established. The variable *wordMarkPace*, indicating the word marking speed, forms the third cluster. According to a diagnosis rule found in de Vicente and Pain [5], fast speed of activity together with high quality of performance (when staying in similarly-difficulty exercises) suggests increasing motivation. Since an increase in the number of words marked is an indication of the student's perceived knowledge (i.e., a reflection of the performance), *wordMarkPace* might be related to the direction of motivation, i.e., Energization.

The tool developed in this study enables to measure online learners' motivation by using solely information stored in log files. However, there are three limitations to this innovative tool. First, variables were identified based on a specific learning environment; it might be useful for similar systems, but for different environments (varied by, e.g., learning domain, instruction modes available) these variables should be converted, and their clustering should be re-examined. Secondly, the classification to the motivation dimensions within the framework has not been validated yet, as well as their actual scales; it is within the authors' agenda to continue in the direction of validation. Third, the tool might not be complete; we only focused on seven variables, however others might be considered. Identifying these variables based on a segment of the learning makes it possible to employ this tool during the learning process; that way, intervention when needed might be possible, and changes in motivation may be analyzed.

Further to the developed tool, the process used in this study – i.e., constructing a literature-based conceptual framework, using *Learnograms* for identifying variables, clustering and classifying these variables within the framework - is of great importance, since it is a procedure which might be transferable to other domains (e.g., anxiety, self-regulation) for developing measuring tools.

Measuring the online learner's motivation has a major role in the instruction-learning cycle. Monitoring the learner's motivation might enable the instructor to interfere when needed (e.g., when student's motivation is decreasing), and should help in developing of

intelligent tutoring systems which react not only to the learner's cognitive behavior but also to her or his affective situation. The overall objective of this underlying approach is to increase the efficiency of the learning process.

References

1. Ames C, Archer J: Achievement goals in the classroom: Students' learning strategies and motivation. Journal of Educational Psychology 80: 260-267 (1988).

2. Beck JE: Using response times to model student disengagement ITS2004 Workshop on Social and Emotional Intelligence in Learning Environments, Maceio, Brazil (2004).

3. Cocea M, Weibelzahl S: Cross-system validation of engagement prediction from log files Second European Conference on Technology Enhanced Learning (EC-TEL 2007), Crete, Greece (2007).

4. de Vicente A, Pain H: Motivation diagnosis in intelligent tutoring systems In: Goettl BP, Halff HM, Redfield CL, Shute VJ (eds) Fourth International Conference on Intelligent Tutoring Systems, pp. 86-95. Springer Berlin, San Antonio, TX (1998).

5. de Vicente A, Pain H: Informing the detection of the students' motivational state: An empirical study The Sixth International Conference on Intelligent Tutoring Systems (ITS 2002), Biarritz, France (2002).

6. Deci EL, Ryan RM: Intrinsic Motivation and Self-Determination in Human Behavior. Plenum, New York, NY (1985).

7. Dweck CS: Motivational processes affecting learning. American Psychologist 41: 1040-1048 (1986).

8. Elliott ES, Dweck CS: Goals: An approach to motivation and achievement. Journal of Personality and Social Psychology 54: 5-12 (1988).

9. Heyman GD, Dweck CS: Achievement goals and intrinsic motivation: Their relation and their role in adaptive motivation. Motivation and Emotion 16: 231-247 (1992).

10. Humphreys MS, Revelle W: Personality, motivation, and performance: A theory of the relationship between individual differences and information processing. Psychological Review 91: 153-184 (1984).

11. Kleinginna PR, Kleinginna AM: A categorized list of emotion definitions, with suggestions for a consensual definition. Motivation and Emotion 5: 345-378 (1981).

12. Masgoret AM, Gardner RC: Attitudes, motivation, and second language learning: A meta-analysis of studies conducted by Gardner and associates. Language Learning 23: 123-163 (2003).

13. Nachmias R, Hershkovitz A: Using Web Mining for Understanding the Behavior of the Online Learner International Workshop on Applying Data Mining in e-Learning, at the 2nd European Conference on Technology Enhanced Learning (EC-TEL'07), Crete, Greece (2007).

14. Qu L, Johnson WL: Detecting the learner's motivational states in an interactive learning environment Artificial Intelligence in Education, Amsterdam, The Netherlands (2005).

15. Ryan RM, Deci EL: Intrinsic and extrinsic motivations: Classic definitions and new directions. Contemporary Educational Psychology 25: 54-67 (2000).

16. Singh K, Granville M, Dika S: Mathematics and science achievement: Effects of motivation, interest, and academic engagement. Journal of Educational Research 95: 323-332 (2002).

17. Zhang G, Cheng Z, He A, Huang T: A WWW-based learner's learning motivation detecting system International Workshop on Research Directions and Challenge Problems in Advanced Information Systems Engineering, Honjo City, Japan (2003).

Mining Free-form Spoken Responses to Tutor Prompts

Xiaonan Zhang[1], Jack Mostow[1], Nell Duke[2], Christina Trotochaud[3], Joseph Valeri[1], Al Corbett[1]
{xiaonanz, mostow, jmv, corbett}@cs.cmu.edu, nkduke@msu.edu, TrotochaudC@glps.k12.mi.us
[1]Project LISTEN, School of Computer Science, Carnegie Mellon University
[2]Literacy Achievement Research Center, Michigan State University
[3]Grand Ledge Public Schools, Grand Ledge, Michigan

Abstract. How can an automated tutor assess children's spoken responses despite imperfect speech recognition? We address this challenge in the context of tutoring children in explicit strategies for reading comprehension. We report initial progress on collecting, annotating, and mining their spoken responses. Collection and annotation yield authentic but sparse data, which we use to synthesize additional realistic data. We train and evaluate a classifier to estimate the probability that a response mentions a given target.

1 Introduction

Speech is the easiest, most natural way for students to respond to tutors. Speech is faster than typing on a keyboard and more expressive than clicking on a menu item. Speech is especially useful for young children because they type slowly and spell poorly. Ideally an intelligent tutor for children would understand their spoken responses to its prompts. Unfortunately, current technology for speech recognition and language understanding has poor accuracy – especially for children's spontaneous speech, which can be difficult even for adults to understand. An intelligent tutor that relies on accurate transcription and interpretation of children's unconstrained speech appears infeasible for years to come. Consequently, the rare intelligent tutors that recognize children's speech constrain it. For example, Project LISTEN's Reading Tutor operates on oral reading of a known text [1].

Thus methods for intelligent tutors to respond effectively to children's unconstrained speech despite imperfect speech understanding could be very useful. We report here on progress toward this goal in the context of a project to teach children explicit strategies for reading comprehension. This project is extending Project LISTEN's Reading Tutor, which listens to children read aloud, so that it also listens to children *think* aloud. This work builds on previous ideas for word spotting (e.g. [2]), confidence annotation in spoken dialogue systems (e.g. [3]), and generating synthetic data (e.g. [4]). This endeavor is relevant to educational data mining in a number of ways. We describe an efficient way to collect authentic student responses with expert tutorial labels. We show how to augment sparse training data by using it to generate realistic synthetic data. Finally, we present empirical evaluations of classifiers trained on this data.

Acknowledgments: The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305B070458 to Carnegie Mellon University, by the National Science Foundation under ITR/IERI Grant No. REC-0326153, and by the Heinz Endowments. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute, the U.S. Department of Education, the National Science Foundation, or the Heinz Endowments. We also thank the educators, students, and LISTENers who helped generate and analyze our data, in particular Ravi Mosur for developing Sphinx-II's acoustic confidence measure.

2 Collecting authentic student and tutor responses

To mine children's spoken responses to automated tutoring on comprehension strategies, we must first collect a goodly amount of data. An obvious approach is to record children's responses to a human tutor. However, this approach suffers from two shortcomings. First, the process is labor-intensive. Second, children respond differently to a human tutor than to an automated tutor, for example because they have a different social relationship to an adult than to a computer. This difference is problematic because our purpose is to enable an automated tutor to assess children's spoken responses. A Wizard of Oz simulation mitigates the social relationship issue, but not the labor-intensiveness.

Instead, we took a different approach. We extended the Reading Tutor by implementing comprehension strategy instruction that prompts and records spoken responses without analyzing them. Children use the Reading Tutor simultaneously on multiple computers, which can record them at the same time, unlike a human tutor or Wizard of Oz limited to tutoring one child at a time. The spoken responses are logged to a database and transcribed by hand.

The instruction itself is scripted by an expert reading researcher and practitioner based on their experience in teaching comprehension strategies to children. They carefully select texts conducive to tutoring particular strategies. Tutorial sequences in the instruction consist of a few basic step types: assisted oral reading by the student; reading aloud by the tutor to the student; multiple choice questions answered by clicking on a menu item; short-answer fill-in questions answered by keyboard input; and prompts for free-form spoken responses.

Our reading experts' involvement did not end with scripting instruction. Once 299 spoken responses to 33 prompts were recorded and transcribed, our expert practitioner annotated each student utterance with how she would have replied to it, and why. An example of a short-answer prompt was *What do you think the fifth sense is?* The expected answer is *touching*. The expert's recommendation for the student responses *touching*, *seeing*, or *use our nose to smell* was to say nothing in reply, since "This question is a "what do you think" question," so any reasonable answer is acceptable. In contrast, the recommended reply to the response *uh, apples?* was *Think about the senses that we already talked about in this text…try again*, because "Apples is not a sense."

Thus our data consists of tutorial prompts, transcribed spoken responses to them, expert annotations that recommend how to reply, and rationales for those recommendations in terms of features of the student responses. The purpose of this data is to train a decision function that uses those features to classify future responses by how the tutor should reply to them. In the examples above, the recommended reply to the student's spoken response depends on whether it mentions a target concept. This type of decision problem is a simple but useful case of the general problem of classifying responses by how the tutor should reply, and applies to many of our expert's annotations. The remainder of this paper focuses on this problem.

3 Detecting a target

We first address how to determine whether a spoken response mentions a given target, which for now we define as a word or phrase plus its variant forms. For example, our target for *touch* includes the forms *touches* and *touching*. We include variant forms because we care if the response mentions the concept, but not which specific word it uses. For the same reason, we plan in future to include synonyms for the context-appropriate word meaning. For example, *feel* and *feeling* are synonyms for *touch* as a sense, but not for the colloquial meaning of *touch* as "ask for money." We also care about our confidence in whether a response mentions a given concept. More precisely, our challenge is to learn the probability that a given response mentions a given target.

3.1 Stretching sparse training data

A difficult learning task requires as much training data as possible. A training example for our task includes an utterance, a target, and a label classifying the utterance by whether it mentions the target. To expand our limited set of authentic data, we generate a much larger set of synthetic data – in fact, so much larger that we hold out authentic data to use for testing. This held-out set consists of the 64 recorded responses to 5 questions where the expected target is clear, e.g. *What do you think the fifth sense is? (touch)*.

Each authentic datum is an annotated utterance labeled as a positive or negative example of mentioning an expected target concept. For example, the utterances *touching* and *uh, apples?* serve respectively as authentic positive and negative examples for the target concept *touch*. We have only a limited amount of such data. To generate a large set of training examples, we reuse the transcribed free-form responses many times as synthetic positive and negative examples of mentioning other concepts. These 471 utterances include 172 free-form responses previously recorded and transcribed but not annotated. The idea is to pretend that each utterance has a different target that it does or does not mention. Thus each utterance generates multiple training examples, one for each hypothetical target. The utterances in the synthetic data are actually authentic; only their labels are not. Thus the utterances *touching* and *uh, apples?* also serve as synthetic examples of mentioning (or not mentioning) hypothetical targets, such as *butterfly*. As this example suggests, the synthetic data is heavily skewed toward negative examples.

As targets for the synthetic data we use the 21 words that occur more than 10 times in the transcribed responses, such as *butterfly*, and include their variants, such as *butterflies*. We exclude the most frequent 100 words of English, such as *the*, because they might differ systematically from authentic target words in how they are spoken. For example, function words tend to have reduced pronunciations. The resulting synthetic data set has $471 \times 21 = 9891$ examples.

3.2 Configuring the speech recognizer

To decide whether an utterance contains a given target, an obvious solution is to use automatic speech recognition (ASR) to decode the utterance, and see if the ASR output contains any of the target words. However, children's free-form speech is too

unpredictable for ASR to transcribe accurately, in contrast to oral reading of known text. How can we configure the ASR to increase its accuracy on this target-spotting task?

A key point here is that if we care only about a specific target, we do not need to know what else the student said. We therefore configure the ASR to listen only for the target words and to insert arbitrary phoneme sequences to model other words. We penalize such insertions to make the ASR prefer target words unless they match the speech poorly.

If this configuration detected the target perfectly, our problem would be solved. However, there are still many cases where the ASR errs. Fortunately, our ASR (http://sourceforge.net/projects/cmusphinx) reports not only which words are recognized, but also an acoustic confidence score for each recognized word. We compute the acoustic confidence for a target concept, e.g., *touch*, as the maximum score in the ASR output of any of the target words, e.g., *touch, touches, touching*. To decide whether the utterance mentions the target, the tutor can test whether this score exceeds some threshold that determines the tradeoff between false negatives and false positives. But can it do better?

3.3 Using a logistic regression model to combine various evidence

The simple acoustic confidence threshold model ignores some relevant factors. A single threshold may not be appropriate for different targets. For example, the larger the set of words or phrases comprising the target, the higher their maximum confidence score may tend to be. If longer words or phrases tend to score lower than shorter ones, the threshold should decrease with target length. Conversely, the longer the utterance, the likelier that it will randomly contain a good match to the target, so the threshold should increase with utterance length. To take these factors into account, we use predictors derived from the utterance and ASR output and listed in Table 1.

Table 1: Predictors used in the logistic regression model

#	Predictor	Description
1	MaxConf	Maximum confidence score of all target words
2	TargetSize	Number of target words (words that belong to the target)
3	WordLen	# of letters in the top-scored target word; if none, 0; if there's a tie, their average
4	HypLen	Length of the ASR output, measured by number of words
5	UttDur	Duration of the utterance, measured by the size of its audio file in kilobytes

To combine this information, we use binomial (or binary) logistic regression, which estimates the probability of an event Y as a logistic function of a set of input predictors $X_1, X_2, \ldots, X_n$. In our case, $Y = 1$ iff a target occurs in an utterance, and $X_1, \ldots, X_5$ are the five predictor variables in Table 1. The logit (*i.e.*, the logarithm of the odds) of the target occurring is modeled as a linear function of the X_i, as shown in Equation 1:

$$\ln\left(\frac{\Pr(occur)}{1-\Pr(occur)}\right) = \beta_0 + \beta_1 * MaxConf + \beta_2 * TargetSize + \beta_3 * WordLen + \beta_4 * HypLen + \beta_5 * UttDur \quad \textbf{(1)}$$

Here Pr(*occur*) is the probability that the target occurs in the utterance, β_0 is the intercept, and β_1, …, β_5 are the respective regression coefficients for the predictors in Table 1. The regression coefficient for each predictor describes the change in the logit associated with a unit change in that predictor. A positive (negative) β means that an increase in the predictor will increase (decrease) the probability of the outcome. To make different β's comparable, we first normalize the input predictors to range from 0 to 1, so that the absolute value of β measures the impact of that predictor compared to the others. Given Pr(*occur*) for a target, we decide whether the target occurs by comparing Pr(*occur*) to a threshold, e.g. 0.5. We decide yes if it's larger than the threshold, otherwise no.

We use a logistic regression model for several reasons. First, it's compact to represent, fast to compute, and easy to interpret. Second, unlike linear regression it does not assume normally distributed variables. Third, rather than a binary judgment as to whether the target occurs, it outputs a probability that a tutor could use to decide more judiciously which feedback to provide. For example, if the tutor thinks the student said the target but is not very confident, it should hedge its reply rather than praise an answer that may well be wrong. Finally, logistic regression outperformed the alternatives we compared it to. In cross-validation tests, it achieved higher precision, recall, and AUC (described in Section 4) than a Naïve Bayes classifier or a J48 decision tree.

We used Weka 3.5.7 (from weka.sourceforge.net) to train the logistic regression model on the 9891 synthetic examples. As noted earlier, the class distribution on synthetic training data is skewed, with 9547 negative examples but only 344 positive examples for the 21 targets defined. In contrast, the 64 held-out authentic utterances are more balanced, comprising 30 positive instances and 34 negative instances. Differences in class distribution between training data and test data can hurt classifier performance, for instance by biasing the classifier against a class rare in the training set but common in the test set. To address this problem, we used Weka's cost-sensitive classification mechanism to balance the training data, so that its distribution of positive and negative instances resembles the distribution on authentic data. Table 2 shows the resulting β parameter estimates for our five predictors.

Table 2: Parameter estimates of the logistic regression model

Predictor	MaxConf	TargetSize	WordLen	HypLen	UttDur
β **value**	9.8659	0.5802	1.5780	2.7986	0.2606

As Table 2 shows, all predictors are positively correlated with the odds that the target occurs, but acoustic confidence is the strongest predictor. Although one might expect long responses to be likelier to contain the target than short responses, the UttDur predictor is very weak, probably because we measured it by the size of the audio

recording. This recording includes the tutor prompt in the background, so its size reflects the combined duration of the prompt and the student's utterance.

4 Evaluation

We tested our logistic regression model on both synthetic and authentic data. We used 10-fold cross validation on the synthetic training data. We also evaluated the model on the 64 authentically labeled utterances we used as held-out test data. We compared against a majority class baseline model, which simply predicts the most common class for all instances. Table 3 compares the model performance on both data sets.

We evaluate the classifiers on several metrics. Overall accuracy is the fraction of cases classified correctly, i.e. (# TP (true positive) + # TN (true negative)) / # total cases, so it reflects the class distribution. The TP rate, also called sensitivity or recall, is the fraction # TP / (# TP + # FN) of actual positive cases correctly classified as positive. The FP (false positive) rate is the fraction # FP / (# TN + # FP) of actual negative cases misclassified as positive. Its complement, called specificity, measures what fraction of actual negative cases is classified correctly as negative. All these metrics depend on the probability threshold for classifying a case as positive – namely 0.5 for our model.

Cross validation of the majority class baseline shows very high accuracy and zero FP rate because the synthetic data is highly skewed toward negative examples; its accuracy is much lower on the authentic data. More importantly, such a classifier is useless because it cannot detect any mention of the target: its TP Rate is 0. In contrast, the logistic regression model is much more sensitive to positive examples.

Table 3: Model performance under different testing options

Testing method	Classifier	Accuracy	TP Rate	FP Rate	AUC
10-fold cross validation	Majority class	96.52 %	0	0	0.496
	Logistic	80.15 %	0.765	0.197	0.867
Test on authentic data	Majority class	54.67 %	0	0	0.5
	Logistic	75.00 %	0.552	0.086	0.796

In practice, for the probabilistic output of logistic regression model Pr(*occur*) to be useful, we need to turn the probabilities into discrete decisions so as to provide tutorial feedback accordingly. For example, if the tutor is very sure that the target didn't occur, it should give corrective feedback; but if it's not sure, then a hedged reply is probably preferable. With this intuition, we decide on a preliminary division of Pr(*occur*) into 3 disjoint regions, based on two threshold values t_h and t_l ($0 < t_l < t_h < 1$):

- Yes: confident that the target occurred in the utterance ($\Pr(occur) \geq t_h$);
- No: confident that the target didn't occur in the utterance ($\Pr(occur) \leq t_l$);
- Unsure: neither ($t_l < \Pr(occur) < t_h$).

These thresholds control the tradeoff between coverage and precision. The higher the value of t_h, the fewer Yes decisions the tutor will make, but the more confident it can be of these decisions (assuming we have a reasonable model). On the other hand, the tutor will hedge more of its feedback, presumably making it less helpful to students.

To describe this tradeoff, Table 4 shows model coverage and precision on the set of 64 authentic responses for various threshold values. In the table, *Pr(Yes)* and *Pr(No)* mean the probability of outputting a Yes and a No decision, respectively. Precision is the proportion of Yes (No) decisions that are in fact correct, i.e., positive (negative) examples. For example, with high_threshold = 0.9 the tutor will decide only about 14% of the time that Yes, the student mentioned the target – but roughly 89% of these decisions will be correct. By dropping this threshold to 0.5, it can decide Yes more than twice as often – almost 30% of responses – and still be right about 84% of them.

Table 4: Model coverage and precision with different threshold values

Deciding Yes			Deciding No		
high_threshold	**Pr(Yes)**	**Precision**	**low_threshold**	**Pr(No)**	**Precision**
0.5	0.2969	0.8421	0.5	0.7031	0.7111
0.6	0.2500	0.8750	0.4	0.5938	0.7105
0.7	0.1875	0.8333	0.3	0.4688	0.7667
0.8	0.1719	0.8182	0.2	0.2500	0.8125
0.9	0.1406	0.8889	0.1	0.1094	1.0000

Table 4 provides guidance both about where to set the threshold values, and about how definitively to phrase tutor feedback. For example, it indicates that precision for Yes decisions is roughly the same (81%-89%) for thresholds from 0.5 to 0.9, so the tutor may as well set high_threshold at 0.5 (possibly even lower) in order to decide Yes more often, but its feedback must reflect that the student response probably contains the target but may well not. For example, the tutor might refrain from confirming the answer as correct, but still treat it as correct in updating its student model. In contrast, precision for No decisions is much more sensitive, ranging from 71% to 100% as low_threshold varies from 0.5 down to 0.1 – but with coverage ranging from over 70% to below 11%. So the tradeoff between coverage and precision differs for the No case. If our authentic training data is representative, setting low_threshold to 0.1 will avoid any false rejections, allowing definitively phrased corrective feedback. However, at this threshold value, the tutor will decide No less than 11% of the time, even though the target will be absent about half the time. On the other hand, a value of 0.5 will let the tutor decide No for 70% of student responses, but only 71% of these decisions will be correct. In this case, tutor feedback must be phrased to avoid characterizing the student response as wrong.

5 Contributions and future work

This paper formulates the general problem of extracting reliable, tutorially useful information from children's free-form spoken responses despite imperfect speech recognition, so as to assess their comprehension and select appropriate tutor feedback.

We focus on the simpler but common and useful case of estimating the probability that an utterance mentions a given target concept.

We describe efficient methods to collect authentic student data labeled by expert tutors, and to expand it into a much larger set of synthetic yet realistic data. We present a logistical regression model to estimate the probability of a target by combining features of the target and utterance with the acoustic confidence output by a speech recognizer. We cross-validate the accuracy of the resulting probability estimates on synthetic data, and evaluate it on a smaller held-out set of authentic data.

Concept mention is just one useful feature for tutors to detect. We need to extend it to handle synonyms, but we have already extended it (in work omitted here to save space) from the single-target problem addressed in this paper to the multiple-target problem of deciding whether an utterance mentions any, all, or none of *N* given targets. Another useful feature is the distinction between confident and tentative responses [6, 7]. Other distinctions in our expert tutor's annotations include correct vs. incorrect, vague vs. detailed, and answered easily vs. with difficulty. Future work includes using these distinctions to update student models and guide tutor decisions.

References

[1] Mostow, J. Is ASR accurate enough for automated reading tutors, and how can we tell? *Proceedings of the Ninth International Conference on Spoken Language Processing (Interspeech — ICSLP)*, 837-840. 2006. Pittsburgh, PA: International Speech Communication Association.

[2] Wilpon, J.G., L.R. Rabiner, C.H. Lee, and E.R. Goldman. Automatic recognition of keywords in unconstrained speech using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1990. 38(11): p. 1870-1878.

[3] San-Segundo, R., B. Pellom, K. Hacioglu, W. W., and J.M.A. Pardo. Confidence measures for spoken dialogue systems. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, p. 393-396. 2001.

[4] McCandless, M. Word Rejection for a Literacy Tutor. In *Department of Electrical and Computer Engineering*. 1992, Massachusetts Institute of Technology: Cambridge, MA.

[5] Duke, N.K. and P.D. Pearson. Effective Practices for Developing Reading Comprehension. In A.E. Farstrup and S.J. Samuels, Editors, *What Research Has To Say about Reading Instruction*, p. 205-242. International Reading Association: Newark, DE, 2002.

[6] Pon-Barry, H., K. Schultz, E.O. Bratt, B. Clark, and S. Peters. Responding to Student Uncertainty in Spoken Tutorial Dialogue Systems. *International Journal of Artificial Intelligence in Education*, 2006. 16(2): p. 171-194.

[7] Forbes-Riley, K., D. Litman, and M. Rotaru. Responding to Student Uncertainty during Computer Tutoring: A Preliminary Evaluation. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS)*. 2008. Montreal, Canada.

Computational Infrastructures for School Improvement: How to Move Forward

R. Benjamin Shapiro, Hisham Petry, and Louis M. Gomez

{rbs, h-petry, l-gomez}@northwestern.edu

Learning Sciences, Northwestern University

Abstract. The instructional practices common in today's schools reveal a disconnect between instruction and evidence of the effects of that instruction on student learning. In this paper, we propose the creation of computational infrastructures that will help teachers make more informed decisions in their practice. These infrastructures formalize student and teacher routines to facilitate data collection and mining, in order to create actionable information. We then show an instance of such a computational infrastructure and describe its potential for improving instruction.

1 Introduction

Studies of today's (American, and many European) schools reveal a disconnect between instruction and evidence of the effects of that instruction on student learning. New constructivist understandings of learning, knowledge, and effective instruction [11] challenge school reformers to instantiate formative assessment and instruction by reconstituting the daily routines of teaching into ones "where a teacher's day-to-day decision-making is instrumentally constructed based on the interaction of detailed observations about students' work in the classroom (and the personal background students bring to their work) and the aims in view for subsequent instruction" [2]. Transitioning the existing American system of education into a system of practice where this occurs could be made more likely by the development of new Computational Infrastructures for School Improvement (CISIs) that help teachers to use, and reflect on their use of, evidence to make decisions. This paper describes a rationale and framework for such architectures, gives a concrete illustration of the implementation of such an infrastructure, and shows how it provides an opportunity to apply Educational Data Mining (EDM) techniques to some of the most challenging problems that teachers face.

This is not fundamentally an empirical paper. Rather, it is an argument for how we could organize the EDM field and the technologies it produces to have higher cumulatively and greater impact on educational practice and research. It is also not a claim about a specific technology; some of the ideas here about feedback and interactive support are manifested in some form in existing work, such as Interactive Tutoring Systems. Instead, we are trying to make a broader claim about how attending to the daily practices of teachers, learners, and school leaders could guide the design of interconnected technologies and practices, allowing EDM to have a much broader impact than it otherwise might.

2 Problem of Practice

Teaching is a complex, difficult, and uncertainty-ridden job. As Higgens notes:

> One starting point for inquiry into the moral phenomenology of teaching is Philip Jackson's famous observation that teachers make over 200 decisions per hour. If Donald Schon is right, all practices require 'reflection in action', but teaching takes the demand for improvisation to new levels. It is in large part this radical unpredictability of teaching which shapes its phenomenology. It does not seem to matter how many times one has taught. Each time one begins a class, there is that unique blend of excitement and dread

> caused by the unpredictability of what will unfold. Perhaps this explains the fetishisation of the lesson plan: we want to deny just how much is unplanned. [5]

In large part it is this uncertainty that makes teaching difficult and inhibits school improvement [9,12], for even the most principled and motivated teacher trying to implement Ambitious Instruction (i.e., trying to make instruction build upon learners' prior knowledge and needs while engaging learners in consequential work; [2]) faces the currently insurmountable challenge of making sense of the volumes of information that are created in the course of daily instructional practice. At the same time, today's information infrastructure through which evidence of past performance flows from class to class and year to year requires that a massive amount of data be discarded on a continuous basis. The rich detail of students' daily work is distilled into grades that provide extremely low resolution; one cannot deduce from a past grade which concepts a student mastered and which he or she did not. Consequently, teachers face at once a glut of temporally local information, too much to completely comprehend, and a dearth of specific information about past performance that could inform present decision making.

Real-time and retrospective EDM, coupled to new architectures and practices for collecting, displaying, and using data, may offer a way past this seeming paradox, leap-frogging the current technology and practice of schooling by helping practitioners to make sense of the volumes of information generated through daily student and teacher practices. A growing number of scholars are discussing data and data use to quantify problems of achievement deficiencies, but few have investigated which data teachers should use and how they should use it to solve problems. Where does the teacher get the most decision-making bang for the data-analytic buck? What problems are students having? Do the problems that students are facing indicate a need to change teaching practice? Are they indicative of gaps in prior understanding? What kind of instruction has helped students with similar misunderstandings solve problems? What are the data that can reliably yield that information in order to support improved teaching? Thus far, most of the tools that support student thinking are either within single domains (such as cognitive tutors e.g., [3,6]) or systems that don't provide information in a way that lets teacher compare across classes and domains to examine their practices (Blackboard, for example, could do this, but does not). We are proposing building integrative architectures that allow information from such systems, used across disciplines/subject areas, over time and across space, to deeply understand students' intellectual development and the instructional activity that is necessary to support it.

Fullan, Hill, and Crevola also perceive instructional technologies to be on the cusp of dramatic improvement [4]. They believe that we now know enough about effective teaching to make large strides through increases in personalization, precision, and professional learning. At the same time that teachers must manage an entire classroom of students, students still have individual skills, proficiencies, and deficits that require more personalized instruction. To respond to those needs, teachers must be precise in their actions, which they suggest requires more precise information. This information could come from data mining, analyzing student work (data) to transform it into action by the teacher, reducing the uncertainty and increasing the precision of practice.

3 Moving From Data to Action

As described above, supporting students' learning is difficult due to the complexity of extracting actionable information from the raw data of student activity. Actionable information is

information that directly supports decision making by entailing specific actions in regard to some problem at hand.

In order to situate technology development (i.e., the design of technologies to create this actionable information) within the broader scope of school improvement support, we conceptualize the work of teaching and learning as sets of routines enacted by intentional agents, facilitated by tools which provide a mechanism for action (agents work through the tools) and are a source of actionable information. Agents do their work with particular goals and knowledges, which are by-products of agents' social/historical contexts and their individual experiences.

Given the right circumstances (e.g., time, need, context...), an expert agent will interpret particular observed information as indicative of specific facts about the nature of the world (including the knowledge of other actors), which have specific implications for action. For example, a particular student response *X* to some task likely indicates some student conceptualization *Y* about the subject matter which has instructional implications *Z*. It is in the sense that from *X* follows *Y* and *Z* that *X* can be said, through the teacher's knowledgeable perception, to be actionable information. And it is because the observing teacher can see *X* as indicative of *Y* and that understanding *Y* warrants *Z* that we can say the teacher is an expert. Another, less expert, teacher might not notice *X* at all or, noticing *X*, might come to some other conclusion *Y'* and action *Z'*. Even the expert teacher, pressed for time or otherwise distracted, might not notice *X*'s significance (its actionability). We believe that tools can encapsulate knowledge (expertise) and, through so doing, can help practitioners to notice more and act more expertly, by guiding action toward Z. Tools that encapsulate this expertise, and help teachers to notice the instructional implications of student work, may help teachers to implement cycles of formative assessment and instruction.

Tools providing actionable information articulate/encapsulate an assembly model by suggesting what the meaning of some pieces of information (raw data) are and what the pedagogical responses to that meaning are (ala *X*, *Y*, and *Z*, above). In order to generate actionable information to guide learning and teaching, our design and evaluation work should include helping teachers to build routines and tools that engage learners in tasks that evoke evidence about their literate and subject-area development and provide actionable renderings of this evidence to guide pedagogical responses to it (i.e., instructional routines informed by the evidence, mediated by the knowledge encapsulated in the tools). Data mining and information visualization techniques form the core of such tools; a crucial empirical challenge, yet to be solved, is exploring the space of such tools to know which can most easily support more informed teacher practice.

4 SPACE, A Prototype CISI

Working with teachers using participatory design methods [7], we have created the beginnings of a CISI, called SPACE. SPACE was originally created to support many of the most information intensive aspects of project-based Inquiry teaching and learning, including teachers' planning, assignment specification, artifact management, formative feedback and assessment, reflection, as well as students' planning, performance, revision, peer critique and assessment. SPACE is a database-backed web application that so far incorporates simple data aggregation and visualization techniques. The case of its design and implementation to support practice at a

middle school in science, social studies, and literacy makes clear two things. First, there is evidence that CISIs support a paradigm shift in teachers' use of evidence for improving instruction and second EDM and data visualization techniques are at the core of what makes CISIs useful. SPACE's underlying information architecture is roughly captured by Figure 1.

In SPACE, project assignments are comprised of ordered sets of task assignments. Each task is subject to a set of assessment criteria, each of which is related to one or more standards of goals (the standards may be State or Federal Standards or local constructs). As students (working alone or in groups) do their work in SPACE, the electronic artifacts they create (the student tasks) are related to the appropriate task template as well as to the authors themselves. Teachers' and students' assessments of work, along with information about (co-)authorship, form a kind of incidental social network, the implications of which we discuss below.

The SPACE user interface makes it easy for students to find out what work they need to do and what criteria will be used to assess it. Teachers and students may easily browse the database to see the work of others and to assess that work. Aggregate representations make it simple for teachers to discover which skills need the most instructional focus. At present, these aggregations are of the most simplistic kind, showing, for each standard or assessment criterion, on a by-student, by-assignment, or by-cohort basis, the mean, standard deviation, and trend over time. Presently SPACE employs relatively simple EDM techniques, but as we discuss next, combining the structured data of a system like SPACE with advancements in EDM could dramatically increase the quality and quantity of actionable information available to teachers.

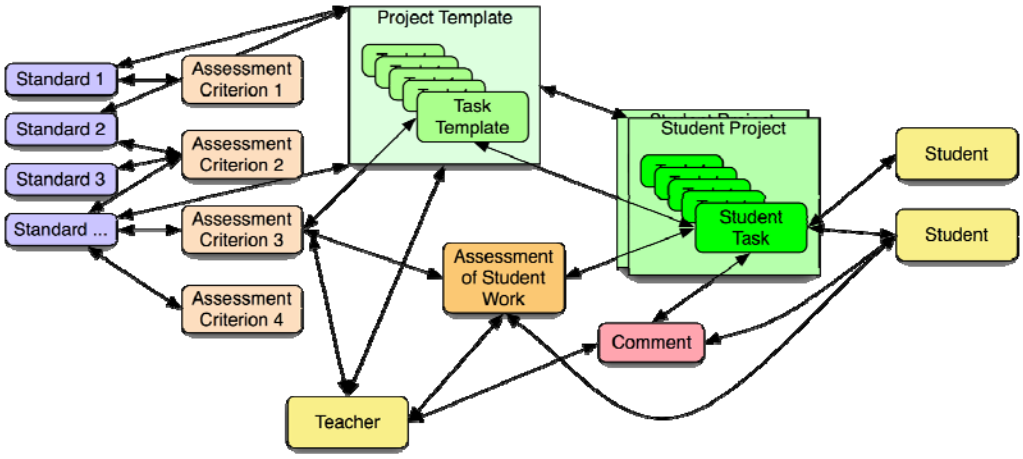


Figure 1: SPACE Information Architecture

5 SPACE in Practice

SPACE was intended to help teachers enact cycles of formative instruction throughout the course of project-based Inquiry. The following example arose during a recent classroom implementation of SPACE supporting a science fair project: Students are working on different projects and they are at different stages of progress. Students submit work to be assessed by their peers and the teacher. The work can then be revised and resubmitted as needed to complete the task. Managing the student schedules is critical because the science fair represents a hard deadline that must be

met. This is a lot of data for a teacher to keep track of, and students (and the teacher) can easily fall behind.

Figure 2 shows a visualization of an eighth grade class midway through a project. The stages represent progress towards the science fair presentation. As an example of what these stages represent, stage one is to read some current events and find something technological that excites the student and take some notes on it (e.g. Nike's new tennis shoe). Stage two is finding a related area of interest (e.g. materials science), and stage three is a project proposal (e.g. comparing springiness of different materials). From there stage ten and eleven are building a data table and collecting data for analysis. Finally, stage 14 and 15 is the write up and presentation of the work. At the time that this figure was generated several students were starting on stage 10.

A tremendous amount of data is being generated: the student work (the pieces of work individually, but also chains of revisions), the critiques, rubric-based assessments, and logs of who's looking at what. This data must be analyzed and presented in a way that is useful to the teacher to become actionable knowledge. Analysis and visualization techniques appraise teachers of students that are waiting for teacher feedback in order to go forward. Other visualizations indicate whether the student is ahead or behind schedule for the science fair.

Looking at this visualization the teacher can know immediately which students are falling behind or need attention. The 'w' indicates tasks that are waiting for teacher feedback. The large ratio of 'w's to non-'w's indicates that this specific teacher is significantly behind in offering feedback, and may be overwhelmed. Moreover, the number of pink and red boxes illustrates that students are struggling to turn work in on time; the number of lagging revision required symbols illustrates that many students have been asked to revise but have not, perhaps because they are confused about what the teacher wants them to do.

The representation in Figure 2 is a high-level view of the underlying data; it foregrounds punctuality and task status, while backgrounding the actual content of the work and assessments thereof. Teachers can click the column headings, author names, or project titles to see all instances of work on a task (across projects), to see information about the authors (including links to other work), or all work on the project, respectively. This representation is only one of many possible ways of aggregating project work. We have already implemented a skill-based aggregator that shows summary statistics about how students' work (at the individual student, project, or cohort level) has been assessed, on a skill-by-skill basis, showing teachers what skills students have mastery of and which they need additional support in. Because all interfaces are massively hyperlinked, it is easy for teachers to find concrete examples of relevant work. We imagine the application of a number of information retrieval and data-mining techniques to provide additional depictions of the data. The following section describes several possibilities.

6 A Way Forward

Singley and Lam describe a number of interesting heuristics that could be helpful to practitioners trying to decide where to direct their attention [14]. Their Classroom Sentinel alerts teachers to conditions, identified through data mining, such as a student's grade dropping significantly from past assessments, a struggling student performing above average on assessments, the student being within range to increase their letter grade if they do well on the next assignment, or ESL students performing below average on a mathematics assessment. [8] describes a framework for data driven decision-making in schools that "highlights three key components: the process by

which raw data becomes useable information, the role of prior knowledge of the decision-maker, and the effect of the data-reporting tool in shaping that process". We try here to describe, in a more concrete fashion, what infrastructure for such decision-making might be.

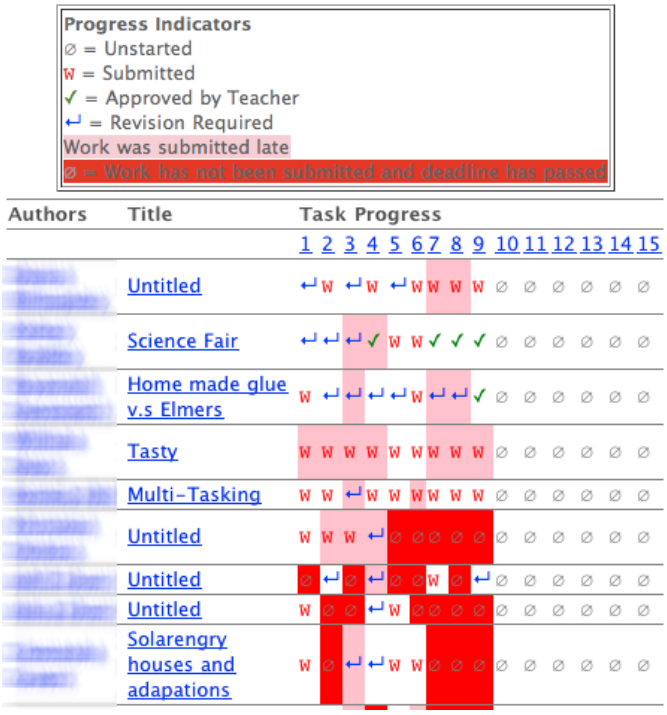


Figure 2: SPACE Project Progress View (for Cohort)

The data structure behind SPACE affords a number of powerful analyses each of which promises to deliver actionable information to teachers and leaders. These analyses range from discovering students' mastery of necessary prior knowledge or related activities, to the leveraging students' social networks, to recommendations of pedagogical options, each of which we now describe.

The SPACE database makes it simple for teachers to connect summary statistics to concrete examples of student work. For example, the Illinois state standards for literacy state that middle school students should be able to "identify appropriate resources to solve problems or answer questions through research"; this standard is relevant to any activity where students must collect sources "from the wild" (e.g., the Internet) in order to do their work. Consequently, in a full implementation of SPACE across a school or district, there would be multiple assignments, in a variety of subject areas, that are related to the standard. Through SPACE, a teacher can see not only summary statistics about students' performances related to the standard, but also concrete examples of work. Teachers could search for all examples (in his/her own class or in others') of poor performance in selecting appropriate articles, or find out which students tend to be poor selectors.

Having this rich data offers interesting opportunities for increasing the (measurement) validity of teachers' assessment practices. Ideally, there would be a high correlation between teachers' assessment of students' work with respect to some set of standards and other measures of those same students' skill with respect to those same standards. For example, if a recent assessment says that a student has trouble summarizing texts, then we'd expect that teacher assessments of

that student's summaries of texts should be similarly low. Divergence between the two measures may indicate a disconnect between teacher understanding of standards and how those standards are measured. Tools like SPACE would allow teachers to take problem areas identified by standardized tests, and then comb through a student's work to verify similar pre-existing issues. This verification process helps teachers identify and recognize problem areas for present and future students. Consequently, tools like SPACE offer the potential to allow teachers to reflect on their practice as well as make standardized tests formative rather than summative assessments.

Education research has found patterns in peer groups and achievement using social network analysis [10,13]. The SPACE database also affords a number of social network analyses. For example, students' critiques of each others' work might represent an edges between students. We can use the amount of subsequent improvement in students' work as the strength of the edges. We can then analyze a network constructed in such a manner over time to understand which student's feedback influences another student (i.e., students' whose critiques are useful to a broad spectrum of peers). This information could be highly actionable for teachers, supporting decision-making about instruction (e.g., knowing who needs help giving good critiques) and classroom management (e.g., which students to pair up so that they'll be maximally mutually supportive).

CISIs have potential beyond what we have described here. Not only could they better support the daily cycles of instruction, but they can also be used to aggregate information at the school level to make decisions. For example they could assist school leaders making decisions about how to allocate discretionary support resources (e.g., teachers' aids, additional training, supplemental money for extra teacher hours). Similar to how teachers have an increased awareness of students' learning, leaders could spot school level problems early on (such as the teacher falling behind in the above example) and intervene to be proactive instead of reactive.

7 Conclusion

Personalizing instruction for students is demanding of teachers. Presently teachers do not have tools that support their daily instructional practice. Computational infrastructures that merge data storage, mining, and presentation can help teachers manage classroom data to make more informed and responsive decisions. Through the use of these sorts of tools, levels of instruction that previously required vast pedagogical content knowledge and heroic effort could now be much more reasonably achieved, with benefits for every student.

8 References

[1] Anjewierden, A., Kolloffel, B., & Hulshof, C. (2007). *Towards educational data mining: Using data mining methods for automated chat analysis to understand and support inquiry learning processes.* Paper presented at the International Workshop on Applying Data Mining in e-Learning, Crete, Greece.

[2] Bryk, A., Gomez, L., Joseph, D., Pinkard, N., Rosen, L., Walker, L. (2006). *Activity Theory Framework for the Information Infrastructure System.* Unpublished Manuscript, Accessed April 7, 2008. http://www.iisrd.org/documents/IIS_Activity_Framework_2006.pdf

[3] Constantino-Gonzalez, M. and Suthers, D.D. (2003). Automated Coaching of Collaboration Based on Workspace Analysis: Evaluation and Implications for Future Learning Environments. *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*.

[4] Fullan, M. G., Crévola, C., & Hill, P. (2006). *Breakthrough*: Corwin Press.

[5] Higgins, C. (2003). MacIntyre's Moral Theory and the Possibility of an Aretaic Ethics of Teaching. *Journal of Philosophy of Education, 37*(2), 279-292.

[6] Kay, J. and Maisonneuve, N. and Yacef, K. and Zaïane, O. (2006). Mining patterns of events in students' teamwork data. *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*, Jhongli, Taiwan

[7] Kensing, F., & Blomberg, J. (1998). Participatory Design: Issues and Concerns. *Computer Supported Cooperative Work (CSCW), 7*(3), 167-185.

[8] Light, D., Wexler, D. H., & Heinze, J. (2004). Keeping Teachers in the Center: A Framework of Data-Driven Decision-Making. *Education Development Center (EDC)*, 6.

[9] Lortie, D. C. (2002). Schoolteacher: A Sociological Study.

[10] Petry, H. J., & Gomez, L. (2008). *On par with peers: Social dynamics of players and workers.* Paper presented at the AERA, New York, NY.

[11] Resnick, L. B., Lesgold, A., & Hall, M. W. (2005). Technology and the New Culture of Learning: Tools for Education Professionals. In P. Gärdenfors & P. Johansson (Eds.), *Cognition, Education, And Communication Technology*.

[12] Rosenholtz, S. J. (1991). *Teacher's workplace: the social organization of schools*: Teachers College Press.

[13] Ryan, A. M. (2001). The peer group as a context for the development of young adolescent motivation and achievement. *Child Development, 72*(4), 1135-1150.

[14] Singley, M. K., & Lam, R. B. (2005). The classroom sentinel: supporting data-driven decision-making in the classroom. *Proceedings of the 14th international conference on World Wide Web*, 315-321.

[15] Stiggins, R. J. (2002). Assessment Crisis: The Absence of Assessment for Learning. *Phi Delta Kappan, 83*(10), 758-765.

A Preliminary Analysis of the Logged Questions that Students Ask in Introductory Computer Science

Cecily Heiner
cecily@cs.utah.edu
School of Computing, University of Utah

Abstract. Asking questions is widely believed to contribute to student learning, but little is known about the questions that students ask or how to exploit them in tutorial interventions to help students learn. This paper presents a preliminary analysis of student questions from an introductory computer science course, logged automatically when students requested help during open lab consulting hours. The data set consists of one hundred and forty two questions that introductory computer science students asked while working on four weekly programming assignments. The initial data suggest that student questions can be repetitive in nature and different students ask different kinds of questions. The paper concludes by suggesting that an analysis technique that is more sophisticated than cosine similarity applied to raw text with stop words removed will be necessary to classify student initiated questions.

1 Introduction

Some students in introductory computer science classes have had little or no prior exposure to programming, and many of their questions are poorly articulated. Students depend on teaching assistants to provide help and feedback. Unfortunately, teaching assistants for introductory computer science classes are a scarce and expensive resource, and the demand for their attention often exceeds their supply of time. These demands come from students who are in the lab as well as students working remotely who send questions electronically.

Previously, students in Computer Science 1 at the University of Utah who wanted help used a Java applet called the TA Call Queue where they entered their name and the location of their lab machine. This information then appeared in the TA interface, and a human TA was required to walk over to the student's machine in order to help them. The TA Call Queue did not allow students to type natural language describing their help request, help students obtain remote assistance, or log data to the server. To improve upon the TA Call Queue, a new piece of software called the Virtual Teaching Assistant was built that logged student questions and allowed for remote feedback when necessary. With both the TA Call Queue and the Virtual TA software, the human TAs periodically circulate through the lab and prompt the students to ask questions because some students are reluctant to ask questions without prompting.

The Spring 2008 version of the Virtual Teaching Assistant student software works as follows. The student decides to ask a question and launches the Virtual TA software by clicking on a button on the class webpage. The application already knows the student login, location, and current assignment number. The student can fill in the name of the Java method and class, and any natural language they need to express their question.

Additionally, the student must attach their source code folder using a standard open file dialog prior to submitting their question. When the student clicks the send button, the client connects to a server, logs all of the information it has collected from the student, and passes the question to a human TA on duty. The student interface is shown in Figure 1a.

The human TA sees the question, source code, and other educational context in a TA interface. In the TA interface, the upper left hand area displays the list of students who have questions on the queue, the lower left hand area lists the source code files associated with the selected student. The center area displays the student source code. The third panel to the right displays the student form and provides an area for the TA to type a response and/or an answer category. An answer category corresponds to a group of questions that could be answered with the same answer. To facilitate the assignment of questions to answer categories, the TA interface has a button panel in the right-most pane with one button for each answer category for the current assignment; the TA can also add a new answer category if the current question does not fit into an existing category. The human TA can answer the question in person by walking to the student's machine in the computer lab, or the human TA can type a text response to the student that will appear in the student's client software in the lower text area. To remove the question from the queue, the TA must assign the question to an answer category. The answer category chosen by the human TA is logged to a database along with any text that the human TA has provided for the student. The TA interface is shown in Figure 1b.

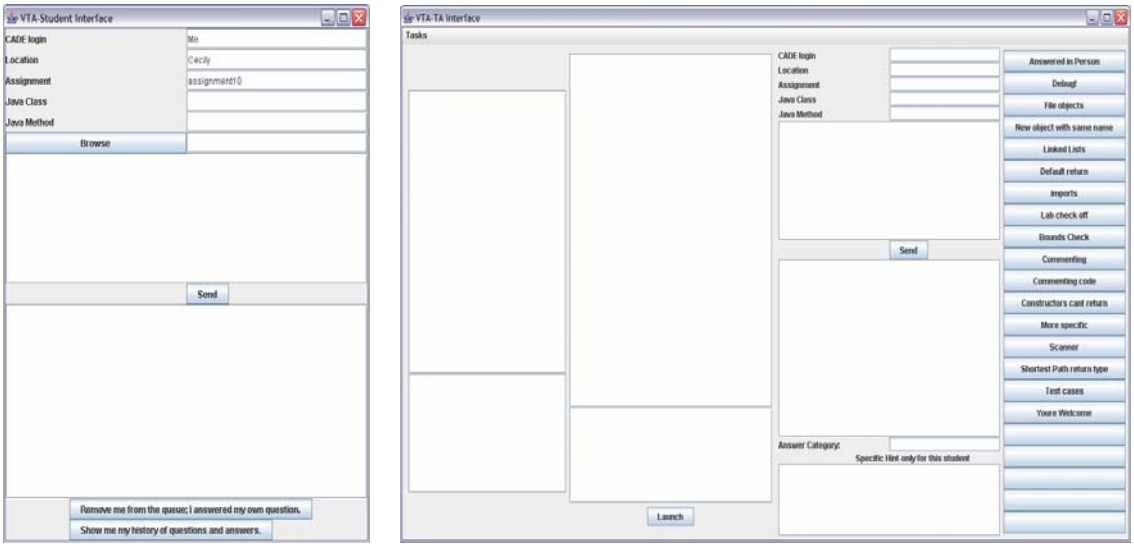


Figure 1. Virtual Teaching Assistant Interfaces for Student(a) and TA(b)

One of the long term goals of the Virtual Teaching Assistant project is to provide automated answers to some of questions that students ask. This paper examines the data logged during one month of usage. The paper presents a preliminary analysis of finding similar questions, and it concludes that applying cosine similarity to raw text with the stop words removed is insufficient for finding similar previous questions that could be exploited in a tutorial intervention.

2 Prior Work

Because novices are notoriously inept at articulating their reasoning, many intelligent tutoring systems use one of two approaches to dealing with the questions that students ask. Several tutoring systems have completely bypassed the problem of poorly articulated student language by restricting the size of the information space that the system covers (e.g. single physics problem[3], a sentence to be read aloud[8], or algebra symbolization[5]), dividing each problem into a set of skills, and creating interventions or help messages for each skill. Alternatively, some systems have focused specifically on helping students to articulate their reasoning (e.g. providing explanations for steps in a geometry problem[2], describing general physics principles in natural language[4], and listing required hardware for a computer task[4]) because generating natural language may help students develop deep learning. However, these systems are generally trying to elicit specific pieces of natural language from the student instead of providing answers to student questions. Research on producing automated responses to student generated questions is remarkably sparse throughout the literature.

3 Study Design

3.1 Participants

The study participants were students from Introduction to Computer Science 1 (CS1410) at the University of Utah. Most students in Computer Science 1 are age 18-22, but there are also a few non-traditional students, such as a student from a local high school or adults who have returned to school later in life. Computer Science 1 is the first required computer science course for computer science majors, and it is a gatekeeper course with a strong emphasis on the Java programming language as well as the traditionally long hours for novice programmers and the typically high dropout, fail, and withdrawal rates. The majority of students who take Computer Science 1 hope to major in computer science or a related field, but they must pass that class along with three others with sufficiently high grades to attain official status as a computer science major. Although approximately seventy eight students were active in the course during the study, only twenty four of them asked questions using the Virtual Teaching Assistant software during the month long study period.

3.2 Data Cleansing

When the data logging software was designed, every effort was made to facilitate rapid data analysis. The long term goal is to complete the analysis for a question in real time and exploit it for an instant tutorial intervention. However, even though the data set was carefully designed, some cleansing was necessary. For example, some students used several computers away from the lab, and their logins needed to be recoded for consistency. In a few cases, the only way to obtain their login was to look in the comments in the source code. Some students asked the same question twice because of a glitch in the software that caused a delay between question submission and system acknowledgement; the duplicate questions were removed from the dataset, but the original questions were left in the dataset. If the human TA did not provide an adequate

answer category for a student question, and a category could not be generated using the student's natural language or the status of the source code, then the question was excluded.

4 Similarity Analyses

To calculate the similarity of a new question to a previous questions, the natural language from the student questions is extracted and represented in vector form where each row of the vector corresponds to a particular word, and the value of that vector element is the number of times that word appears in the student's question. The vector representation excluded stop words from this list[1]; stop words are words such as "the", "a", "I", "you", and other common words. A vector similarity measurement calculates the similarity of a pair of questions on a scale of 0 to 1. In these measurements, 0 means that the pair has no common words, and 1 means that the questions are identical. The experiments described below utilize cosine similarity[7].

As a gold standard, a question is similar to a previous question if it has the same answer category. Table 1 provides an illustrative data set. To conserve space, questions and answer categories are referred to by number; in the actual analysis, the original natural language is used. As described in the introduction, when using the Virtual TA system, the human TAs must assign an answer category to each student question that they answer. A previous question with the same answer category is called a *target question*. In Table 2

Table , "target question(s)" lists previous questions with the same answer category. Additionally, for each question, the similarity between that question and each previous question is calculated. The previous question that has the highest similarity score when paired with the current question has the "max similarity score", and it is considered the "most similar previous question". If multiple previous questions have the same highest similarity score, the most recent question is considered the "most similar previous question".

Table 1. Question Similarity Scores

	Answer Category	Target Question(s)	Q1 Sim.	Q2 Sim.	Q3 Sim	Q4 Sim	Q5 Sim	Q6 Sim	Max Similarity Score	Most Similar Previous Question
Q1	A1									
Q2	A2		0						0	Q1
Q3	A3		0	0					0	Q2
Q4	A1	Q1	0.47	0	0				0.47	Q1
Q5	A4		0	0	0	0			0	Q4
Q6	A4	Q5	0	0.03	0	0.12	0.1		0.12	Q4
Q7	A5		0.1	0.05	0.05	0.04	0	0	.05	Q3

The original plan was to simply use the answer categories as recorded by the human TAs. Unfortunately, not all of the answer categories that the TAs chose are particularly descriptive of the student's question. For example, the human TAs often used the category "Answered in Person", as opposed to a category that provided a meaningful description of the student's question. Excluding such poorly described answer category data results in a smaller dataset as shown in the "Excluding recoded data" and "Excluding both" columns of Table 2. A second alternative is to recode the data; resulting in a larger dataset as shown in the "Everything" and "Excluding compiler errors" columns of Table 2. This disaggregation shows how much of each type of data was recoded.

An additional disaggregation considers the questions associated with source code that doesn't compile. Because the compiler can indicate without human intervention whether or not the source code compiles, the kinds of automated data collection and analyses that are possible differ depending on whether or not the source code compiles. For example, if the source code does not compile, the compiler output may be exploited for question classification. If the source code does compile, then comparison algorithms can exploit the output of parse tree analysis or runtime analysis techniques such as program slicing and comparison[6].

Table 2. Data set with and without exclusions

	Everything	Excluding recoded data	Excluding compiler errors	Excluding both
Total Questions	142	87	80	53
Target Question Exists	35	13	11	6
Target Question Found	6	3	4	2

In all four conditions shown in Table 2, the algorithm found similar previous questions for at least a few questions, but only a very small percentage of the time. Similar previous questions could be exploited in a tutorial intervention if a similarity score threshold separates those questions with a target question from those questions without a target question. Figure 2 is an attempt to find such a threshold when excluding compiler errors. The independent variable is the maximum similarity score for a question, and the dependent variable 0 if there was not a similar previous question, 1 if there was a most similar question and it was found by the algorithm, and 0.5 if there was a similar previous question, but it was not found by the algorithm. If Figure 2 were a step function, then the threshold would be the location of the step. Unfortunately, this data does not seem to have such a threshold. Consequently, applying cosine similarity to raw text with stop words removed is insufficient to classify many student questions, and maximum similarity score with natural language is insufficient to identify previous questions to exploit in an automated tutoring intervention.

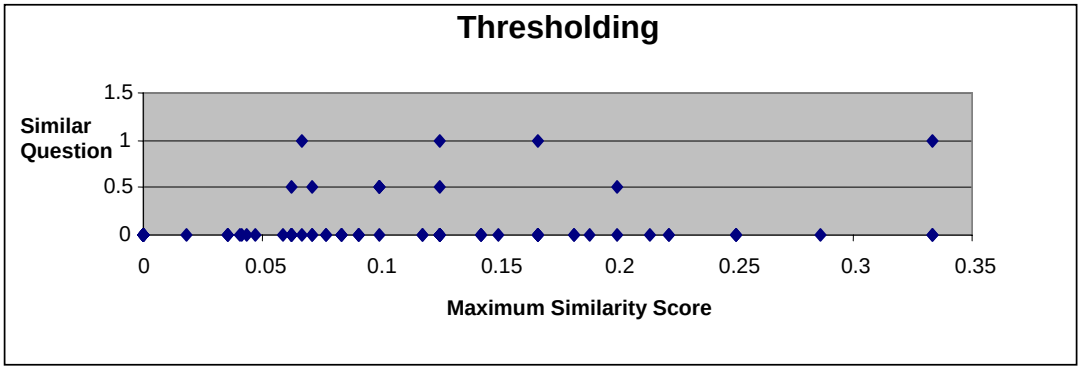


Figure 2. Thresholding

4 Conclusions

4.1 Limitations

This work has all of the standard limitations of work based on a single intelligent tutoring system deployed in a single class; the results hold for this population of students using this tutoring system in the setting described in the paper. Additionally, the numbers are small because it is a only month long study. The paper does not show p-values for any of the claims, nor does it calculate the correlations of variables. A first attempt at correlating the number of questions with average assignment score or midterm, not presented in this paper, suggests that this is a difficult problem for this domain because several of the students who ask questions early in the semester decide to drop, withdraw, or fail before the first midterm. How to properly deal with such students is an interesting, open research question for educational data mining.

4.2 Contributions

This is the first paper to analyze data from the Spring 2008 version of the Virtual Teaching Assistant, a tool that logs the questions that students ask and the answer categories that human TAs use to describe student questions. A set of preliminary analyses suggest that using raw natural language is not sufficient to find previous student questions that can be exploited in an automated tutorial intervention.

4.3 Future Work

This paper presents a preliminary analysis of a month-long study of student questions using only the natural language of the questions that students asked. This analysis may be refined using a more complete task-specific stopword list. Several words such as “cool”, “awesome”, “thanks”, “need”, “help”, and “question” that are not normally considered stopwords behave like stop words for this task. Treating them like stopwords may improve accuracy. Additionally, a more vigorous recoding of answer categories and/or a clearer set of guidelines for creating answer may improve question classification accuracy. Even in the recoded categories, there are several redundant answer categories

such as “Moving the pyramid”, “sizing the pyramid”, and “Pyramid location” that should probably all be recoded to the same general category; another example is the redundant answer categories of “mortgage equation” and “calculating mortgage”.

Previous work suggests that the majority of questions asked by students about a programming assignment are often clustered around one or two topics[9], but these topics are too vague to be useful for question classification; creating a question classification scheme that is simple enough for introductory computer science TAs to use is an area that merits further research.

The next step for this research is to cleanse the data more thoroughly and scale up this study to a semester long study, and include other forms of data such as student source code and educational context, e.g. the assignment the student is working on, as part of the classification algorithm. The plan is to compare the plain natural language version with the extended version that uses source code and educational context, and hopefully show that using the extra data results in a statistically significant improvement in question classification. Also, with improved question classification, it may be possible to exploit the question classification in a tutorial intervention to help students learn more. Showing that previous questions, mined in real time from the knowledge base of the system, can be used in a tutorial intervention to help students learn is the long term goal of this research project.

In addition to extending the VTA project, future work is needed to clarify the differences in student and tutor initiated open ended dialog. Previous work has shown that cosine similarity was as effective as an untrained human tutor in a tutor initiated open ended dialog[10], but in this scenario, the student can exploit both the natural language in the tutor-initiated question as well as text within a window from which the question was drawn. Consider the example given in that paper: The tutor-initiated question is “How is an operating system like a *communications* coordinator?”, a student response is “It *communicates* with the peripherals”, and a model answer is “A computer's operating system includes programs that take care of the details of *communication* with peripherals.” (emphasis added) As the emphasis added shows, many terms that are used in a tutor-initiated question also tend to appear in both the student answer and the model answer. The topic of this question was probably drawn from a specific segment of a written text, and with high probability the student would have extracted their answers from a segment of the text with the same information. Student initiated questions do not have the added advantage of the language from the tutor-prompt, and it is not clear whether or not cosine-similarity can be exploited to help answer them. A more detailed comparison of several dialogues from both student-initiated and tutor-initiated systems may shed light on additional differences in dialogues based on who initiates them.

Acknowledgements

Thanks to Hal Daume, Joe Zachary, Joseph Beck, Peter Jensen and others who have provided helpful comments and feedback on various versions of this work.

References

[1] cited 2008; Available from: http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words.

[2] Aleven, V., Popescu, O., Koedinger, K. Pilot-Testing a Tutorial Dialogue System That Supports Self-Explanation. *Intelligent Tutoring Systems*, 2002. Springer Berlin / Heidelberg.

[3] Gertner, A. S., VanLehn, K. Andes: A Coached Problem Solving Environment for Physics. *Intelligent Tutoring Systems*, 2000. Springer.

[4] Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., Harter, D., Intelligent Tutoring Systems with Conversational Dialogue, in *AI Magazine*. 2001. p. 39-50.

[5] Heffernan, N., Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of Human Tutors, in PhD Thesis. 2001, Carnegie Mellon University: Pittsburgh, Pennsylvania.

[6] Jensen, P., Hybrid Automated Fault Localization in Programs written by Novice Programmers, in PhD Thesis. 2007, University of Utah: Salt Lake City.

[7] Manning, C. D., Schutze, H., *Foundations of Statistical Natural Language Processing*. 1999, Cambridge, Massachusetts: The MIT Press.

[8] Mostow, J., Aist, G., 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.

[9] Robins, A., Haden, P., Garner, S. Problem Distributions in a CS1 Course. *Australian Computing Education Conference*, 2005. Conferences in Research in Practice in Information Technology.

[10] Wiemer-Hastings, P., Wiemer-Hastings, K., Graesser, A. C. Approximate natural language understanding for an intelligent tutor. *12th International Florida Artificial Intelligence Research Conference*, 1999. AAAI Press.

Reinforcement Learning-based Feature Selection For Developing Pedagogically Effective Tutorial Dialogue Tactics

Min Chi, Pamela Jordan, Kurt VanLehn, Moses Hall
{mic31, pjordan+, vanlehn+, mosesh}@pitt.edu
Learning Research Development Center & Intelligent System Program,
University of Pittsburgh

Abstract. Given the subtlety of tutorial tactics, identifying effective pedagogical tactical rules from human tutoring dialogues and implementing them for dialogue tutoring systems is not trivial. In this work, we used reinforcement learning (RL) to automatically derive pedagogical tutoring dialog tactics. Past research has shown that the choice of the features significantly affects the effectiveness of the learned tactics. We defined a total of 18 features which we classified into four types. First, we compared five feature selection methods and overall upper-bound method seems to be most efficient. Then we compared the four types of features and found that temporal situation and autonomy related features are significantly more relevant and effective to tutorial decisions than either performance or situation related features.

1 Introduction

One challenging issue confronting dialogue tutoring systems is how to select the best tutoring actions for any student state so as to maximize learning gains. We call this decision making *tutoring tactics* since it generally governs brief episodes of tutoring dialogue, such as a single step, and seems to be crucial for achieving further improvements in pedagogical effectiveness. Bad tutorial decisions have the potential to bore or frustrate students to the point that they fail to learn. Many existing dialogue systems use hand-crafted tactical rules derived from analyzing human dialogues [6,13]. However, identifying and implementing these tactical rules is not trivial. It has been shown that expert human tutors generally employ a range of tutoring actions and their decisions depend in subtle ways on the student's competence, student's self-confidence and other factors. For example, one important tutoring tactic is whether the tutor should *tell* the student the answer directly for a step or whether he should *elicit* the step from the student with a prompt or a series of questions. Collins et al. suggest that when a student is unfamiliar with the target knowledge, the tutor should tell it directly; when a student becomes more familiar with the knowledge, the tutor should elicit the answer via questions; and once a student has mastered the knowledge, it doesn't matter whether it is told or elicited [4,5].

Most existing dialogue tutoring systems with hand-crafted rules react sensibly when students exhibit less than optimal behaviors; however, their tutorial interactions are much stiffer than those of human tutors and they have not yet achieved the same effectiveness that one-on-one, face-to-face expert human tutors have [1]. In recent years, work in designing spoken dialogue systems has proposed several data-driven methodologies; one of them is Reinforcement Learning (RL) [3,12,13]. It has been shown that RL is effective at automatically learning the best action to take at any state in a dialogue. And such success inspired us to use RL for dialogue tutoring systems.

Our work can be divided into three stages. In the first stage, we built an initial tutorial dialogue system and collected an exploratory corpus by using the system to train 64 students. In this initial system, decisions on certain types of tutoring actions were made randomly. In the second stage, we used RL on the exploratory corpus to derive tutoring tactics for these actions and then incorporated the learned tactics into the initial system. The modified system is identical to the initial one except that the tutoring actions that were previously made randomly are now based on

the learned tutoring tactics. In the last stage, the modified system is being used to train a new group of students so that we compare their learning performance to that of the students previously trained on the initial system. We expect the modified system to be so well attuned to students that it will be more effective than the initial system.

One tactical decision investigated in this study was whether the tutor should *tell* the next step to the student or *elicit* it from the student. We defined 18 features to model the dialogue and students' state. They were selected based upon the four types of features that were shown in [7] to be relevant for human tutors to making their tutorial decisions: autonomy, temporal situation, situation, and performance respectively. Among 18 features, features 1-3 are autonomy related based upon the amount of the work that the tutor has let students do; features 4-8 cover time-related information such as time spent on the training so far; features 9-11 are situation related including whether the student and tutor are now problem solving or in post discussion, the difficulty level of the problem and so on; and finally, features 12-18 cover the performance information such as the correctness of student's previous answers and the ability of student.

In an RL model, the size of the state space increases exponentially as the number of involved features increases. In order to learn effective tutoring tactics, we should have a corpus that covers each of these states at least once, which means $\geq 2^{18}$ in our case. However, it is almost impossible to do so given the high cost of collecting educational data. On the other hand, the learned policy may become too subtle to be necessary. Figure 1 shows an example of a learned policy involving five features: [1, 2, 4, 15, 16], in which feature 2 is an autonomy feature defined as the percentage of elicitation the students have received so far. Each of the five features was converted from a real number to a binary value based upon their median scores, for example, f2='.4982' means if feature 2 value is above .4982, it is 1 otherwise, it is 0. There were a total of 32 rules learned: in 10 situations the tutor should elicit, in 19 it should tell; in the remaining 3 cases either will do. For example, when all of the features are 0 then the tutor should tell as 0:0:0:0:0 the first in the list of tells. As you can see, five features provide a large space is already quite complex and is already much more subtle than most of the tutorial tactics derived from analizing human turorial dialogues [4,5].

```
KC22    'features'=[1, 2, 4, 15 ,16],
        'cutoff'=[f1='1.0000' f2='.4982' f4='56.0000'  f15='.6154'    f='.2683'  ],
        'policy':
          'elicit: [0:0:0:0:1, 0:0:0:1:0, 1:0:0:0:0, 1:0:0:0:1, 1:0:0:1:0, 1:1:0:1:1,
                    0:0:1:0:0, 0:0:1:0:1, 1:0:1:0:0, 1:0:1:0:1],
          'tell:   [0:0:0:0:0, 0:0:0:1:1, 0:1:0:0:0, 0:1:0:0:1, 0:1:0:1:0, 0:1:0:1:1, 1:1:0:0:1,
                    1:1:0:1:0, 0:0:1:1:0, 0:0:1:1:1, 0:1:1:0:0, 0:1:1:0:1, 0:1:1:1:0, 0:1:1:1:1,
                    1:0:1:1:1, 1:1:1:0:0, 1:1:1:0:1, 1:1:1:1:0, 1:1:1:1:1]
           'else: [1:0:0:1:1, 1:1:0:0:0, 1:0:1:1:0]
```

Figure 1. A Learned Policy Based On Five Features On KC22

Based on the size of the exploratory corpus we collected in the first stage, we decided that no more than *six* features should be used. Previous research involving RL and feature selection in dialogue systems either focused on selecting features with certain characteristics [2,7,8] or investigated a relatively small number of features [10,11]. Therefore, in this study, we proposed four general RL-based feature selection methods.

2 Background

Past research on using RL to improve spoken dialogue systems has commonly used Markov Decision Processes (MDP's). An MDP [8,9] describes a stochastic control process and formally corresponds to a 4-tuple (S,A,T,R), where $S = \{s_i\}_{i=1..m}$, is a finite set of process states; $A=\{a_k\}_{k=1..m}$ is a finite set of actions; $T : S \times A \times S \rightarrow [0, 1]$ is a set of transition probabilities between states that describe the dynamics of the modeled system; for example: $P_t(s_i|s_j, a_k)$ is the probability that the model would transition from state s_j to state s_i by taking action a_k at time t; and $R : S \times A \times S \rightarrow R$ denotes a reward model that assigns rewards to state transitions and models payoffs associated with such transitions. The goal of using MDPs is to determine the best policy π^*: the set of actions the model should take at each state s_i to maximize its expected cumulative utility (V-value), which can be calculated from the following recursive equation:

$$V(s_i) = \sum_{s_j} P_{s_i s_j}^{\pi^*(s_i)}[R_{s_i s_j}^{\pi^*(s_i)} + \gamma V(s_j)]$$

As long as a proper state space, action space and reward function are set up, an MDP allows one to automatically derive and compute the best policy. For dialogue tutoring systems, deriving effective tutorial tactics from tutoring dialogues can be naturally cast in the MDP formalism: for the state space, each s_i can be viewed as a vector of features representing the tutoring dialogue context, the student's knowledge level and so on. Action space corresponds to tutoring actions, e.g. elicit or tell; the reward function corresponds to students' learning gains. For tutoring dialogues, the reward is a *delayed* reward because for each state the reward is not known until students have completed the tutoring and taken the post-test.

In this work, we used an MDP package that was designed for Tetreault & Litman's studies since it has proven to be both reliable and successful [10,11]. In previous studies, Tetreault & Litman primarily investigated methods for evaluating whether certain features would improve policy effectiveness. There are two main differences between our study and theirs: firstly, they used a previously collected corpus that was not exploratory with respect to tutorial tactics to train an MDP model in that the tutor often used only one type of action in many dialogue states, which severely limited the types of questions that they could investigate[10,11]; while we use a more exploratory corpus by training students on a dialogue tutoring system in which multiple actions can often be taken and random tactical decisions were made. Thus it is better suited for creating an MDP. Secondly, they did not need to address the problem of general feature selection methods since they only used five features while we had to select up to six of 18.

To evaluate the learned policies, we use three criteria: Expected Cumulative Reward (ECR) and lower-bound and higher-bound of the 95% confidence interval. ECR is the average reward one would expect in the MDP and it is calculated by normalizing the V-value of each state by the number of times it occurs as a start state in a dialogue and then summing over all states [11]. The confidence interval represents how reliable the ECR of a learned policy is [10]. The wider it is, the less reliable the policy's ECR is.

For example, for a learned policy ***A*** derived from feature 2 alone on definition of spring potential energy (KC22) is "if the percent of elicit so far is less than 49.82%, tutor should elicit otherwise the tutor should tell."; ***A*** has ECR = 3.02 (range [-100, 100]) with a 95% confidence interval= [-2.71, 8.45], which means there is a 95% chance that the ECR of the learned policy is between a lower-bound of -2.71 and an upper-bound of 8.45. In Figure 1, also on KC22 but involves five features: 1, 2, 4, 15 and 16. It has ECR = 44.29 with a 95% confidence interval= [23.49, 50.51], which is much more effective than policy ***A*** because even its lower-bound is much higher than policy A's upper-bound.

Sometimes we encounter situations in which the ECR for A is the same as for B; but the confidence interval of A is much narrower than that of B. By only using the three criteria described above, policy A and B cannot be compared. Therefore, we define the hedge of a learned policy as:

$$Hedge = \frac{ECR}{(\{upper-bound\} - \{lower-bound\})}$$

By using the hedge, we can say that policy A is better than policy B since the hedge of A is higher than the hedge of policy B.

3 Experiment

The domain chosen for this study is work and energy covered in college physics. The procedure was as follows students: (1) read a short textbook, which describes the major principles and concepts; (2) take a pre-test; (3) work through seven open-ended training problems with a dialogue tutoring system (Cordillera [14]); and (4) take a post-test that is identical to the pre-test. In the stage one, 64 college students who had not taken any college physics completed the experiment, receiving payment for their participation. Students needed 8-15 hours to complete the procedures and usually required 4-6 sessions of about 2 hours each. Thus, the collected corpus comprises 64 human-computer tutoring dialogues and each dialogue is an extended interaction with one student that covers seven different college-level physics problems. The number of state-action pairs in each of the 64 collected tutorial dialogues varies from 700 to 900.

All training problems, and all pre- and post-tests problems were selected from 127 quantitative and qualitative problems collected from various physics literature. In order to solve these 127 problems, 32 unique knowledge components (KCs) were identified as necessary. A KC is "a generalization of everyday terms like concept, principle, fact, or skill, and cognitive science terms like schema, production rule, misconception, or facet" [14]. For example, KC22[1] is both a concept and a principle while KC23[2] is a fact. KC22 consists of both procedural knowledge and declarative knowledge while KC23 is mainly declarative knowledge and thus learning these two KCs clearly involves different cognitive skills. Therefore, for different KCs, we expect that different features should be considered when making a tactical decision and different tactical decisions should be derived.

In order to learn KC-specific tutorial tactics, students' pre- and post-test scores were also calculated per KC. It turned out our training, pre- and post-problems covered 27 of the 32 possible KCs. Three of the 27 KCs showed up in the tutoring dialogue but were not associated with any action decision points; five KCs coincided only once or twice with decision points; and one KC did not appear in the pre- and post-tests. Therefore, we were left with 18 KCs for which it is possible for us to derive tutoring tactics. Comparisons of pre- and post-test scores indicated that students did learn during their training with Cordillera: their post-test scores were significantly higher than their pre-test scores: $t(126)= 3.36$, $p= 0.001$. Similar to Tetreault & Litman's work, we used students' normalized learning gains (NLGs) for each KC_i as the *delayed* reward function, which is defined as:

$$NLG(KC_i) = \frac{Posttest(KC_i) - \Pr etest(KC_i)}{1 - \Pr etest(KC_i)}.$$

[1] If an object and a spring are in a system, then their spring potential energy is 0.5* k*d^2, where k is the spring constant and d is the displacement of the object relative to the equilibrium position of the spring.
[2] The unit for energy is the Joule.

For each KC_i, the 64 students were split into two groups by the median value of their NLG on it, so the better-performing students' dialogues were given a positive reward of +100, while the other half were assigned a negative reward of -100. The rewards were assigned in the final dialogue state.

Additionally, we annotated our tutoring dialogues and action decisions based on which KCs a tutor action or tutor-student pair of turns covered; Table 1 shows an example. Here lines 1, 5, 6, 7, 9 and 11 are tutor turns, the action type and their target KCs are listed on the right. For example, in the first utterance is an elicit action regarding KC22 while the fifth is a tell action directly stating KC22. Lines 2 and 12 are student responses and they are both correct (as indicated by the '+'); row 2 is about KC22 while row 12 is about KC27.

Table 1. Examples of Tutorial Dialogue with KC Labels

1.	T: At T2 we also know the spring is uncompressed. So, what type of energy can we easily find now?	elicit, KC22
2.	S: spe	+KC22
3.	T: Yes, I agree.	
4.	T: At this point we know the values for KE2 and GPE2. It is time to compute the value for SPE2.	
5.	T: It is important to remember that we cannot find the SPE of an OBJECT. We need to find the spring potential energy of a pair: object-spring.	tell, KC22
6.	T: It is possible to compute the spring potential energy at T2 because we know the spring is uncompressed. That is, d2=0 m.	tell
7.	T: Using this information we can infer that the spring potential energy of the block-spring pair at T2 is SPE2=0 J.	tell, KC22 KC23
8.	T: Now we have all we need to…	
9.	T: TME of the system at T2 is the sum of the block`s KE and the block-Earth pair`s GPE because the value for the spring potential energy is zero. In this case TME2=11.88 J+1.62 J =13.5 J.	tell, KC23 KC24
10.	T: At this point we have found the value for TME2. We can now easily find the system`s TME at all other time points since the system is isolated.	
11.	T: What principle gives us the system`s total mechanical energy at every time point, knowing its value at T2 and also given that the system is isolated?	elicit, KC27
12.	S: student conservation of energy	+ KC27
13.	T: Yes.	

4 Methods

The four feature selection approaches we propose in this paper are fairly straightforward and share the same procedure, which consists of the three phases described below.

Phase 1: For each of 18 features in the state space, use MDP to get a single-feature-policy.
Phase 2: Sort all of the features based on the learned single-feature-policies from ***high to low*** by one of the following measures:
- ECR
- Lower-bound
- Upper-bound
- Hedge

Phase 3: Start with the first feature in the sorted feature list, add one feature at a time to the MDP and learn a new policy. Repeat this process repeat 5 times.

Based on the sorting criteria in phase 2, we named our four feature selection methods: ECR, lower-bound, upper-bound, and hedge respectively. Since these values were calculated from the single-feature-policies learned using MDP in phase 1, the four methods are RL-based. We expect them to be more effective than a random selection. To test our expectations, we created 120 random policies by running the MDP package on randomly selected features (20 rounds for each randomly selected feature set, where each set contained between one and six features). Therefore, for each KC and each feature set size, we learned one tutoring tactic for each RL-based method plus 20 for random feature selection. This gave us (1*4+20)*6 =144 policies.

For each KC_i, we selected one policy that had the highest ECR, lower-bound and upper-bound from the 144 learned policies and named it the "best policy". To quantitatively evaluate how much less effective a learned tutoring tactic is than the best policy, we defined a normalized ECR (NECR) for a learned tutoring tactic as:

$$\mathrm{NECR}(\mathrm{KC_i}, \mathrm{N}, \mathrm{Method_k}) = \frac{1}{C}\sum_{C}\frac{\mathrm{ECR}(\mathrm{KC_i}, \mathrm{N}, \mathrm{Method_k}) - \mathrm{Min\_ECR}(\mathrm{KC_i})}{\mathrm{Max\_ECR}(\mathrm{KC_i}) - \mathrm{Min\_ECR}(\mathrm{KC_i})}$$

$$Max\_ECR(KC_i) = \max_{N\in\{1..6\}, me\{all\ methods\}}(ECR(KC_i, N, Method_m)$$

$$Min\_ECR(KC_{ij}) = \min_{N\in\{1..6\}, me\{all\ methods\}}(ECR(KC_i, N, Method_m)$$

$\mathrm{Max\_ECR}(\mathrm{KC_i})$ and $\mathrm{Min\_ECR}(\mathrm{KC_i})$ are the maximum and minimum ECR among all 144 of the learned policies for KC_i. C is a constant with C=1 for each of the four RL-based methods and C=20 for random feature selection. The maximum NECR for a learned policy is 1 if the learned policy is the best one and the minimum is 0 if the learned policy is the worst one.

5 Results

In order to evaluate the feature selection methods and how the feature effectiveness, we have two main goals. First, we will compare the four RL-based feature selection methods against random feature selection; second we will investigate which features seem to be more important for deriving the best tutoring tactics for deciding to elicit/tell across all KCs.

5.1 Comparing four RL-based methods against random feature selection

Table 2. Comparing the Average NECR of Five Selection Methods for Increasing Feature Set Sizes

Number Features Involved	Upper-bound	ECR	Hedge	Lower-bound	Random
1	0.345	0.372	0.337	0.355	0.119
2	0.355	0.370	0.335	0.314	0.196
3	0.416	0.447	0.400	0.352	0.275
4	0.550	0.515	0.520	0.419	0.348
5	**0.682**	0.579	0.573	0.435	0.422
6	0.673	0.614	0.594	0.485	0.480

NECR is defined as how much a learned tutoring tactics is less effective than the best policy. Table 2 shows the average NECR given the number of selected features and feature selection

method across all 18 KCs. As expected, random feature selection has the worst average NECR regardless of the number of features involved. Overall, if the number of features ≤3, then the ECR approach is the best feature selection method; but if the number of features is between 4 and 6, then the upper-bound method is the best. As the number of features involved increases, the effectiveness of the learned policies for elicit/tell tends to become better, except for upper-bound, which has a better average NECR when there are five features than when there are six. Overall, using the upper-bound method to select five features seems to be the most effective method across all KCs on elicit/tell action decision.

On the other hand, across all 18 KCs the number of times that each method found the best policy is: fourteen times for random, three for upper-bound, two for ECR, one for hedge and none for lower-bound method. The total did not add up to 18 KCs because for KC1, three methods, ecr, hedge, and upper-bound all found the same best policy. Therefore, although random feature selection has the worse average NECR than other four RL-based methods, it is an effective way to find best policies. However, note that the random feature selection method was repeated 120 times for each KC and so it has a total of 120*18=2160 chances to find the 14 best policies; while the upper-bound method was applied only 6 times for each KC and thus has a total of 108 chances. Additionally, because our state space is still relatively small, we expect that the performance of random selection would decrease significantly as the number of features in the state space increases. However, for the four RL-based feature selection methods, increasing the number of features would not decrease their effectiveness since they do not directly depend on the number of features in the state space.

Moreover we compared the best policy learned for each KC by the upper-bound method and those learned by random selection. Over all 18 KCs, the upper-bound policies with just 108 attempts were only 9.46% less effective than random feature selection policies with 2160 attempts. These results indicate that that in education, features that may result higher learning gains should always be always be considered by the tutor when making decisions. This is likely due to the fact that in the worst case a student will simply not learn rather than lose information so the cost of incorporating superfluous feature is low..

5.2 Frequency of Features in Best Policies

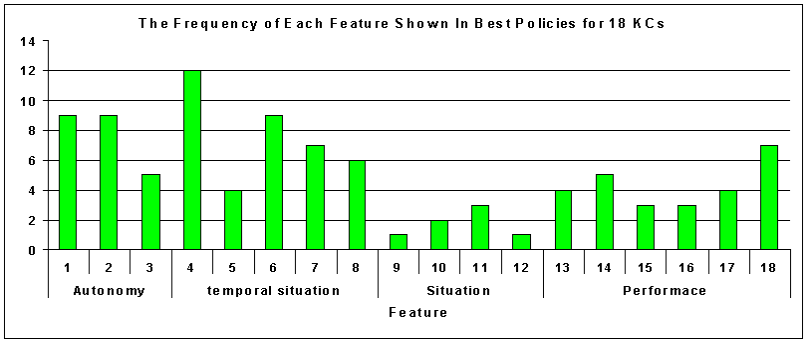


Figure 2. The Frequency of Each Feature Shown In Best Policies.

Figure 2 shows the frequency with which each feature appears in the best policies. This frequency differs significantly among the four feature types: $F(3)= 7.47$, $p=0.003$. There is no significant difference between the three autonomy and five temporal situation related features. When combined they are significantly more frequent than either the performance or situation related features: $t(12)= 2.74$, $p=0.18$ and $t(10)= 4.26$, $p=0.002$ respectively. Consistent with previous research based on analyzing human tutorial dialogue, autonomy related features seemed to be more relevant in deriving effective tutorial tactics. Additionally, we found that temporal

situation related features were also relevant, even more so than the performance related ones when deciding whether to elicit or tell. This was not indicated in previous literature on human tutorial dialog analysis. One possible explanation for this was that in most of the prior literature, temporal situation related factors were often not considered.

6 Conclusions and Future Work

In this paper, we described our work on applying RL methods to derive effective tutoring tactics for elicit/tell. We showed that deriving effective tutoring tactics from tutoring dialogues can be cast in the MDP formalism. Additionally, we proposed four RL-based domain-general feature selection methods. We found the upper-bound method to be more effective than the others.

One of our goals for future work is to investigate how to still get reasonable policies without annotating individual KCs in the dialogues. The annotation process is prohibitively time-consuming and it is not unusual for domain experts to disagree [14]. Another of our goals for future work is to determine how to learn one reasonable policy for all KCs without sacrificing too much of the expected effectiveness. Another two important issues are how to avoid the expensive initial data collection and how to combine the new data with the existing data so that we can learn even more powerful policies.

Acknowledgments
We would like to thank the NLT group and Diane J. Litman for their comments. Support for this research was provided by NSF grants #0325054.

References

[1] B. S. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring.," Educational Researcher, 1984. no. 13, p. 4–16,
[2] M. Frampton and O. Lemon. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*, 2005.
[3] J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In IJCAI Wkshp. on K&R in Practical Dialogue Systems, 2005.
[4] Collins, A., and Stevens, A Goals and Strategies of Inquiry Teachers: In R. Vlaser (Ed.) Advances in Instructional Psychology, 1982. vol. 2 Hillsdale, NJ: Erlbaum Assoc. p. 65-119.
[5] Collins, A.. Design issues for learning environments. In S. Vosniadou, E. D. Corte, R. Glaser, & H. Mandl (Eds.), International perspectives on the design of technology-supported learning environments, 1996. p. 347-361. Mahwah, NJ: Erlbaum,
[6] M. Evens and J. Michael, One-on-One Tutoring by Humans and Computers, Lawrence Erlbaum Associates, Inc., 2006.
[7] Johanna D. Moore, Kaska Porayska-Pomsta, Sebastian Varges, Claus Zinn: Generating Tutorial Feedback with Affect. FLAIRS Conference, 2004.
[8] S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems. In *Proc. NIPS '99*, 1999.
[9] R. Sutton and A. Barto. *Reinforcement Learning*. The MIT Press, 1998.
[10] J. Tetreault, D. Bohus, and D. Litman, "Estimating the reliability of mdp policies: a confidence interval ap proach," in *NAACL*, 2007.
[11] J. Tetreault and D. Litman. Comparing the utility of state features in spoken dialogue using reinforcement learning. In *NAACL*, 2006.
[12] J. Williams, P. Poupart, and S. Young. Factored partially observable markov decision processes for dialogue management. In IJCAI Wkshp. on K&R in Practical Dialogue Systems, 2005.
[13] M. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. JAIR, 12, 2000.
[14] VanLehn, K, Jordan, & Diane. Developing pedagogically effective tutorial dialogue tactics: Experiments and a testbed. In proceedings of SLaTE Workshop, 2007.

Do Students Who See More Concepts in an ITS Learn More?

Moffat Mathews and Tanja Mitrović
{moffat, tanja}@cosc.canterbury.ac.nz
Intelligent Computer Tutoring Group, Computer Science & Software Engineering,
University of Canterbury

Abstract. Active engagement in the subject material has been strongly linked to deeper learning. In traditional teaching environments, even though the student might be presented with new concepts, it is possible for the student to remain passive to such an extent that it is detrimental to learning. This research explores whether experiencing new concepts in an ITS necessarily equates to learning. Initial analysis of data mining student models in SQL-Tutor, a CBM tutor, shows a strong positive correlation between the number of constraints seen and the number of constraints learned. This global trend is mitigated at an individual level, possibly due to individual differences in learning style and behavior. The number of constraints not learned remains relatively constant for all students; however, the proportion of constraints not learned is inversely proportional to the constraints seen. The author suggests deeper analysis into the factors that might cause variability amongst individuals from this population trend.

1 Introduction

For many years, researchers have been stating the importance of active participation and engagement in learning [1, 2]. Constructivists would argue that for effective learning to occur, the student has to be actively involved in constructing the appropriate mental model of the domain, ideally guided by the teacher [3]. Learning which requires the student to actively construct their knowledge leads to better performance (both in time and the number of important errors) [4]. Students who work at constructing their own knowledge using methods such as self-explanation, become better problem solvers [5]. In contrast, students that passively sit in a traditional class environment learn poorly compared to those that are actively involved in constructing their knowledge [6].

It is easy to understand the scenario of a student who learns very little in a traditional lecture environment even if many new concepts are introduced, because they are disengaged from their learning (e.g. daydreaming). However, does this same principle apply in ITSs? Can students use an ITS and learn nothing because they are so passive with their learning? Basic observation would suggest that there are differences between the way a student learns in a traditional lecture environment and within an ITS. In the traditional lecture environment, the student can be passive (in spite of being presented with new concepts) to the extent that they learn nothing. For example, an extreme case might mean they could have slept through the lecture. The progress through material, including the rate at which it is delivered, is usually dependent on an external source (e.g. the lecturer) rather than the student. However, in an ITS, the student generally must initiate all progress i.e. to be presented with anything, the student has to do something. This might be as small as selecting a new problem or requesting help. Disengaging totally

from the learning would mean not progressing through the ITS. There usually is no external entity built into the ITS that controls and maintains the steady flow and delivery of knowledge. Progress is student-dependent. Due to this basic difference, one could assume that if a student progresses through problems in an ITS (in any manner), they are at least in some way actively involved in some part of their learning. As active participation is correlated to learning, students in an ITS must learn more as they progress through the system.

The aim of this research is to mine student logs in an ITS to see if students learn more as they see and experience more concepts or domain principles in the ITS. The goal of this paper is to complete the first stage of this research i.e. to use simple statistics to find general trends within populations. Once these trends are established, in-depth analysis can be performed.

In the next section we introduce the ITS used, namely SQL-Tutor. After describing the dataset, we outline the methods used to perform this analysis. We finish with a discussion of the results and future work.

2 SQL-Tutor

The data used for this research was gathered from SQL-Tutor [7], an Intelligent Tutoring System. It provides students with intelligent and adaptive guidance as they practice their skills within a specifically designed problem solving environment for the domain of database querying using Structured Query Language (SQL). It has been used by students in tertiary database courses since 1998. Having undergone several evaluation studies, it has been shown to produce high levels (both rates and depth) of learning [8].

The domain in SQL-Tutor is modeled using constraints. Constraints are domain principles. They contain a relevance condition and a satisfaction condition. The relevance condition is the condition in which the constraint is applicable (or is relevant). The satisfaction condition states what must be true for this constraint to be correct. For example, for the domain "driving", one constraint might state "if you are driving in New Zealand, you must be on the left-hand side of the road." The relevance condition informs us on when this constraint applies: *"if you are driving in New Zealand"*. If this constraint is relevant, then for it to be correct, the satisfaction condition must be true, i.e. *"you must be on the left-hand side of the road"*. See [9] for a detailed explanation of Constraint-Based Modeling.

Each time a student attempts a problem in SQL-Tutor, the constraints that are relevant for the attempt are recorded in the student model. For each relevant constraint, a history of correct usage for each attempt is also recorded.

SQL-Tutor contains approximately 280 problems. Each problem is assigned a difficulty level by the teacher, using their expertise in the domain. Difficulty levels range from 1 (easiest) to 9 (hardest). For each solution to a problem, several constraints could be relevant i.e. each problem could relate to several concepts. The number and complexity of the relevant constraints also determines the difficulty of the problem. For example, a

problem with many relevant constraints, or a problem with complex relevant constraints could be said to contain many principles or complex principles respectively, making the problem more difficult than one with fewer, simpler constraints. This means that when solving more difficult problems, the student could be dealing with either a large number of constraints or more complex constraints.

3 Data Used

The data for this analysis was collated from student models and logs from an online version of SQL-Tutor. Students from all over the world were given free access to this version of SQL-Tutor when they bought certain database textbooks. Students that made less than five attempts were excluded from this analysis. The final dataset consisted of 1,803 students who spent just over 1,959 combined hours while making a total of 104,062 submissions. In total, these students solved just over 70% (19,604) of the problems they attempted (27,676).

Certain constraints (such as basic syntactic constraints) are always relevant for every query i.e. those concepts apply to every problem. As these constraints are principles that are very easily learned and do not provide in-depth knowledge into the domain or its concepts, they were excluded from this study. The number of constraints that students saw at least once while solving problems varied from 6 to 333. While engaged in their problem solving activities, students experienced a combined total of 174,309 constraints.

4 Method

The method utilized was very simple. We first extracted constraint histories from individual logs and student models. Using this data, we counted the number of unique constraints that were relevant for each student. These are the constraints that the student saw or experienced during their practice; we listed these as "constraints seen".

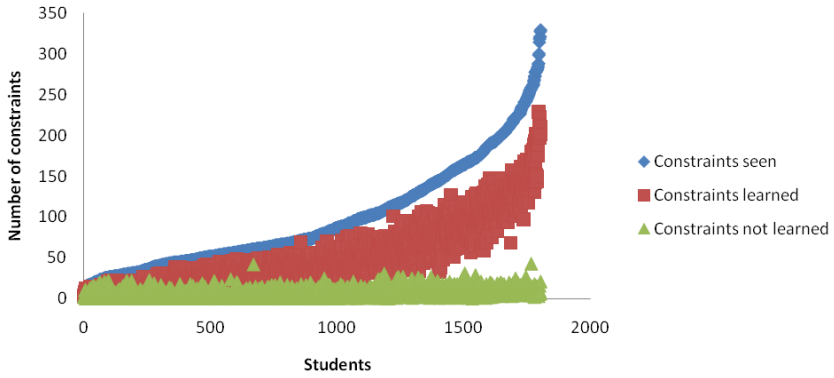


Figure 1: Constraints seen, constraints learned, and constraints not learned for each student, ordered by constraints seen.

To calculate whether a constraint was learned, we used two separate methods. In both methods, we used a window to focus on the recent usage of each constraint. We used the most recent history as we determined that this would better indicate the current state of

learning with regards to that particular constraint. The main difference between the two methods was the size of the window.

In the first method, if the total history of a particular constraint consisted of five or more attempts, we used a window size of five. If the history consisted of less than five attempts, we used a window size of three. In the second method, we always used a window size of five. Each method makes an assumption of how many attempts are required to determine the state of learning for each constraint.

We then calculated the proportion the constraint was learned by dividing the number of correct usages of the constraint in the window by the window size.

$$Proportion\ constraint\ learned = \frac{Correct\ usages\ of\ constraint\ in\ window}{Window\ size}$$

This gave us a number between zero and one, where zero meant that the constraint was not learned at all whereas one meant that the constraint was learned. All other numbers in between zero and one showed the proportion the constraint was learned. We plotted graphs for all users depicting their number of constraints seen, the number of constraints learned, and the number of constraints not learned, using both the methods described above. Both methods above gave very similar graphs. Figure 1 shows the values from method 2. We used these methods as they were previously used somewhat successfully in various versions and evaluation studies.

We also calculated the average difficulty level of problems attempted and problems solved for each student. This is shown in Figure 2.

The total time spent on the system by each student was recorded and graphed against the number of constraints seen (Figure 3). Furthermore, we calculated the *time per constraint seen* and the *time per constraint learned* for each student (Figure 4). Time per constraint seen was calculated by dividing the total time by number of constraints seen. Time per constraint learned was similarly calculated by dividing total time by the number of constraints learned. These calculations were to give us an idea of how much time each student spent in relation to the constraints they had seen (how quickly they were progressing through the system) and to the constraints they had learned (how quickly they were learning constraints).

5 Results and Discussion

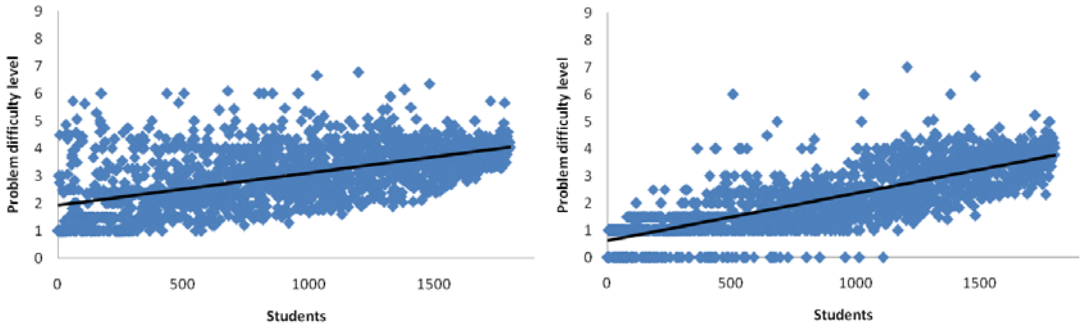


Figure 2: Average difficulty levels of problems attempted (left) and solved (right) for all students ordered by number of constraints seen.

5.1 Constraints

As seen in Figure 1 and from correlation calculations, the number of constraints a student sees is strongly correlated to the constraints learned ($Pearson's\ r = 0.947$). This means that the students who saw more domain principles learned more. This supports our original hypothesis that learning on an ITS requires some form of active engagement which translates to learning. Remember, here we are saying that to have *seen* a constraint, they must have made at least an attempt where that constraint was relevant. Simply moving through and viewing the problem is not counted as progressing through the system as the student would not have experienced any constraints i.e. they have not made any attempts. The variability in the constraints learned could be attributed to individual differences. These differences could be due to the quality and quantity of engagement, the amount of help used, gaming [10], and low skill levels (e.g. low meta-cognitive abilities).

The number of constraints not learned remains relatively constant for all users, regardless of the number of constraints seen. This is interesting, but makes sense as the proportion of constraints that are wrong decreases as the number of constraints seen increases. However, it seems that the types of constraints that are not learned are different between users. Even though all students are getting approximately the same number of constraints wrong, the students who have seen more constraints are dealing with more difficult constraints. This can be seen from the graphs in Figure 2. Students who had low numbers of constraints seen (to the left of each graph) are generally attempting and solving easier problems compared to those who have seen higher numbers of constraints (to the right of each graph). As mentioned earlier, this means that the students on the right of the graph are using more complex constraints or a greater number of constraints in a single problem than the ones on the left.

5.2 *Time*

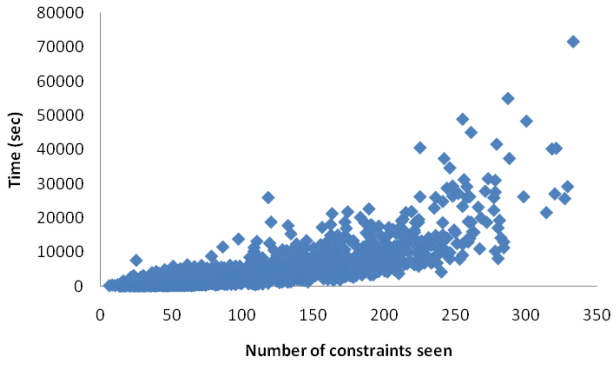


Figure 3: Number of constraints seen against total time taken by students

Figure 3 shows the number of constraints seen plotted against the total time spent in the system. From the graph, we can see that the total time is proportional to the number of constraints seen i.e. the students that saw more constraints spent longer in the system. This is understandable as the greater the number of concepts explored by the student, the longer they will spend in the system. Although this may seem a trivial point, it is included for completeness.

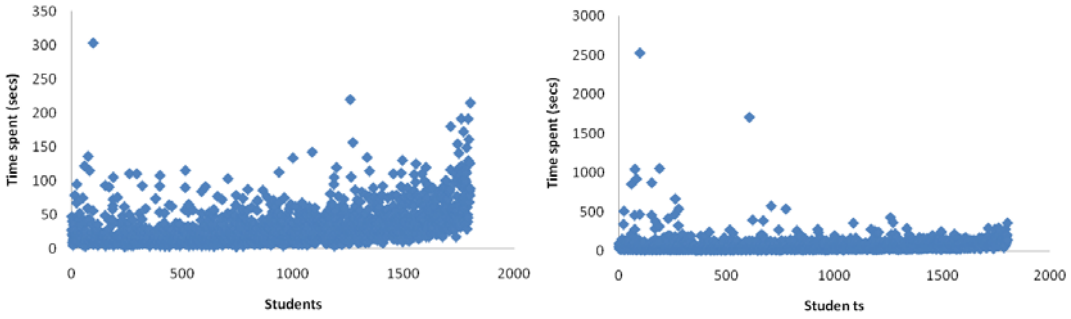


Figure 4: Time spent per constraint seen (left) and constraint learned (right) for each student

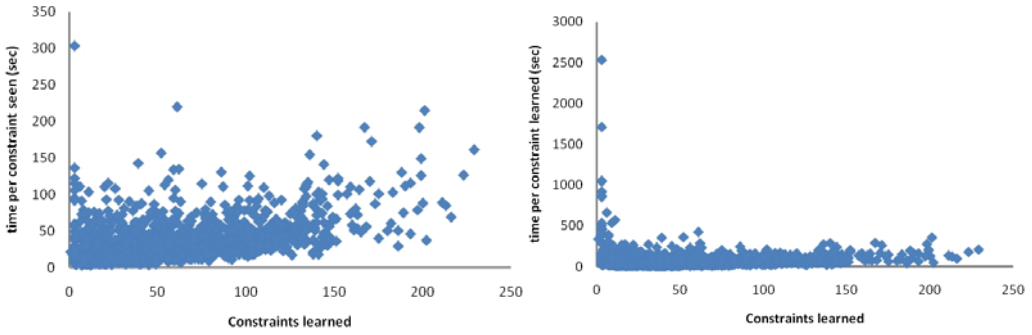


Figure 5: Time spent per constraints seen (left) and constraints learned (right) against constraints learned

The time spent per constraint seen and the time spent per constraint learned (Figure 4) is relatively constant across students, irrespective of the number of constraints seen or learned. In Figure 4, the students who have seen the least number of constraints are towards the left of the graphs. This means that even for the relatively difficult constraints, students on average still spent the same amount of time on each constraint. This could be because students that are seeing more constraints are generally the ones that have progressed to higher levels of expertise. The outliers could be students who are attempting more difficult problems than their expertise level. Similarly, Figure 5 shows that the time spent per constraint seen is relatively constant (or increases slightly with high variability) as more constraints are learned. The time per constraint learned is relatively constant as the number of constraints learned increases. This means that although students spent varying amounts of time on the ITS (depending on the number of constraints they saw), they spent on average the same amount of time per constraint.

6 Conclusion and Future Work

This paper looked at the broad global trends within a population of students using an Intelligent Tutoring System (SQL-Tutor). On a global (population) level, there is a strong positive correlation between the number of constraints a student sees and the number of constraints they learn within an ITS. This seems to support our initial hypothesis that those students who progress through problems in an ITS do learn concepts as they require at least a certain amount of active participation. However, at a more localized level, there are students who have seen many constraints but have learned far fewer constraints than their counterparts who have seen fewer constraints. This could be due to individual differences in the way they learn (e.g. their styles of learning) or in their behavior (e.g. gaming the system). More detailed analysis is required to understand what factors play a part in this process, including factors that affect causality.

We would like to perform deeper analysis on the types of constraints that the students do not learn. What concepts in SQL do they not understand? Are there similarities between these constraints? Are there certain constructs that are inherently more difficult to such a point that no students understand them? Are students guessing their answers and therefore seeing constraints that are not necessarily relevant for the problems they are solving? The answers to these questions would give us better insight into the domain and the way in which students learn concepts.

Another part of this research which requires further exploration is the method by which we calculate constraints learned. In our research, we used two separate methods which gave very similar results. However, *"what does it mean to have learned a constraint or principle?"* is still a very valid question. How much of the constraint history should we use to determine if the constraint has been learned?

In the introduction, we made a comparison between the student in the traditional classroom environment and a student working on an ITS. We said that the extremely passive student in a traditional classroom might not learn anything while the student working on an ITS has a higher chance of learning concepts as they see more concepts. However, it should be noted that the extremely passive student might be one that lacks

motivation to use the ITS. In fact, we did not include anyone who made less than five attempts in our dataset. In this research we do not make any mention about motivating students to learn; merely the trends found in students that do use ITSs.

In spite of these points, this initial study is encouraging, indicating the effectiveness of learning for students that progress through the material in ITSs.

References

[1] R. S. Prawat, "Teachers' Beliefs about Teaching and Learning: A Constructivist Perspective," *American Journal of Education*, vol. 100, pp. 354-395, 1992.

[2] F. N. Akhras and J. A. Self, "System Intelligence in Constructivist Learning," *International Journal of Artificial Intelligence in Education*, vol. 11, pp. 344-376, 2000.

[3] M. Ben-Ari, "Constructivism in Computer Science Education," presented at SIGSCE, Atlanta, GA, USA, 1998.

[4] R. Catrambone and M. Yuasa, "Acquisition of procedures: The effects of example elaborations and active learning exercises," *Learning and Instruction*, vol. 16, pp. 139-153, 2006.

[5] M. T. H. Chi and K. A. VanLehn, "The Content of Physics Self-Explanations," *The Journal of the Learning Sciences*, vol. 1, pp. 69-105, 1991.

[6] L. B. Resnick, "Constructing Knowledge in School," in *Development and Learning: Conflict or Congruence?*, L. S. Liben, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 19-50.

[7] A. Mitrović, "An Intelligent SQL Tutor on the Web," *International Journal of Artificial Intelligence in Education*, vol. 13, pp. 173-197, 2003.

[8] A. Mitrović, B. Martin, and P. Suraweera, "Intelligent Tutors for All: The Constraint-Based Approach," *IEEE Intelligent Educational Systems*, vol. 22, pp. 38-45, 2007.

[9] S. Ohlsson, "Constraint-Based Student Modelling," in *Student Modelling: The Key to Individualized Knowledge-based Instruction*, vol. 125, J. E. Greer and G. I. McCalla, Eds. Berlin: Springer-Verlag GmbH, 1994, pp. 167-189.

[10] R. S. Baker, A. T. Corbett, K. R. Koedinger, and I. Roll, "Detecting When Students Game the System, Across Tutor Subjects and Classroom Cohorts," presented at UM 2005, Berlin, Heidelberg, 2005.