Wong, L.-H. et al. (Eds.) (2013). Proceedings of the 21st International Conference on Computers in Education. Indonesia: Asia-Pacific Society for Computers in Education

Predicting Students' Performance and Problem Solving Behavior from iList Log Data

Omar ALZOUBI [a*], Davide FOSSATI [a], Barbara DI EUGENIO [b], Nick GREEN[b] & Lin CHEN[b]
[a]*Carnegie Mellon University in Qatar, Qatar*
[b]*Computer Science , University of Illinois at Chicago, USA*
*oalzoubi@cmu.edu

Abstract: In this paper, we analyze data gathered from students' interactions with iList, an intelligent tutoring system that teaches linked lists to computer science (CS) undergraduates. A number of features have been extracted from the log files which were used to; a) build predictive models of students' performance, b) analyze temporal aspects of students' problem solving behavior. Our results suggest that it is possible to build predictive models of performance with an accuracy of 87% by using logistic regression. The results also show that it is more likely a student will perform a step correctly if s/he spends more time on it.

Keywords: ITS log file, data mining, intelligent tutoring system, educational data

1. Introduction

Educational data mining (EDM) research focuses on analyzing data gathered from students' interactions with computer-based educational systems. It aims at answering educational research questions and developing methods to better understand students and the settings which they learn in (Baker & Yacef, 2009). Mining log data of intelligent tutoring systems (ITS) has the potential of revealing information important to students, educators, and developers of these systems. Information that could make education more effective and responsive to individual needs (Mostow & Beck, 2006). Log files of an ITS generally contain information related to users' performance and problem solving behavior while using the system. Data mining techniques can be applied to information extracted from log files to build models that can predict future performance of users (Cetintas, Si, Xin, & Hord, 2010; Romero, Ventura, Espejo, & Hervás, 2008). This is an important step for adapting system behavior and for providing proper interventions (e.g., feedback) to enhance learning.

The ability to predict students' performance is very important in an educational environment. It has been suggested that improved student performance prediction could save students tremendous amount of time and effort that could be alternatively used to learn other tasks (Cen, Koedinger, & Junker, 2006). Previous research has utilized log data to design predictive models of users' performance. For example, Cetintas et al. (2010) used data gathered from log data to build a model that was able to predict correctness of problem solving. Their model utilized some performance, problem, time, and mouse movement features to build the model. Similarly, Beck and Woolf (2000) used traces from previous users of a mathematical tutor to construct a linear regression model that predicts if a student will be able to answer a problem correctly, and how long it will take to answer. Their model was very accurate at predicting the time students required to generate the response, and was somewhat accurate at predicting the likelihood of students answering correctly.

In this study we use data gathered from students' interactions with the iList ITS, which we will introduce later. We focus on predicting students' performance using a number of features extracted from the log files. The contributions of this paper are two fold; first, we apply data mining and statistical techniques to iList log data in order to understand problem solving behavior of students. This eventually will lead to improved system design. Second, we build predictive models of students' performance from features extracted from the log files. Building high-level student models from log data has the advantage of not explicitly using any expert knowledge of the domain. The rest of the paper is organized as follows. Section 2 gives an overview of the iList intelligent tutoring system. Section 3 describes our methodology of data collection and features extracted from log data. Section 4 presents and discusses our results, and describes future work directions.

2. The iList System

iList is an intelligent tutoring system that is designed to aid teaching basic concepts of data structures to CS students, particularly linked lists (Fossati et al., 2009). The iList system provides students with a simulated environment where linked lists are represented graphically and can be manipulated using programming commands (C++ or Java). Students are asked by the system to solve certain problems such as, insertion or deletion of a node into a linked list, or performing more complicated operations. A problem represented by an initial state space is presented, students are then asked to perform a number of steps to modify the state space till the desired configuration is reached. The system supports two types of problems; the first type is one where the student delivers a solution in a step-by-step fashion. The other type requires the student to write an entire block of code that typically involves constructs such as *for* loops. In this study we focus only on the former type of problems.

The main components of iList include: 1) the constraint evaluator which checks the correctness of the submitted solution. The usage of constraints in iList is motivated by constraint-based-modeling where domain knowledge is modeled as set of constraints. In a linked list problem, there are several properties that a solution should have in order to be correct. For example, lists should be free of cycles; lists should not terminate with undefined or incorrect pointers; and no nodes should be made unreachable from any of the variables. With these properties represented as constraints, iList can catch many common mistakes students make. 2) The procedural knowledge Model (PKM) is an important architectural feature of iList. The model is built from past log files of students' interactions. The core of this model is a probabilistic graph equivalent to a Markov Chain. Its main components are states and actions. A state "S" corresponds to a possible linked list state, whereas an action "A" corresponds to a command that can modify that state. In the graph, actions and states are represented as vertices. Frequencies associated with states and actions are converted into probabilities using maximum likelihood estimation. The "goodness" value ("G") is then computed for each state of the graph, and it represents a lower bound on the probability that a student will reach a correct solution from that state. Additionally, for each state, "criticality" ("C") is computed which represents the probability that a student will get into a hopeless state (a state with goodness G = 0) from the current state. At run time student actions can be matched against the graph. And 3) the feedback manager interacts with both units in order to generate and issue feedback messages as students are progressing towards the solution.

iList uses both positive and negative feedback strategies to guide learners through their learning activity, which essentially depends on the correctness of students' responses. Negative feedback is given in response to students' mistakes, which allows students to learn from them. According to Ohlsson (1996) people indeed learn from making mistakes and correcting them. On the other hand, positive feedback is provided in response to students making correct steps. This type of feedback can reinforce the correct knowledge students have, or allow them to successfully integrate new knowledge if this correct input was originated by a tentative step (Ohlsson, 2008). Negative feedback is a predominant feature generated by many ITSs today, as those systems are designed to react to students' mistakes. However there is increasing evidence that suggests positive feedback may also be important in enhancing students' learning (Fossati et al., 2009). We can describe the five main types of feedback generated by the feedback manager in iList:

Syntactic feedback – Is generated when commands entered by students are syntactically incorrect.

Execution feedback – The system successfully parses the command entered by student, but encounters a runtime error, e.g. reference to a non-existing variable.

Final feedback – Is presented when the student explicitly asks the system to evaluate his/her solution. Final feedback comes from a collection of feedback units associated with the individual constraints that have been violated, which in turn highlight gaps or incorrect knowledge.

Reactive feedback – This feedback is presented in response to a student move, and it depends on the goodness of a move and its level of uncertainty. Both factors can be obtained from the PKM. For example, a student enters a command that gets correctly executed, but the student has not yet completed the problem. This type of feedback can be either negative or positive feedback.

Proactive feedback – iList can look forward to moves a student can make at any given time and it can intervene with this type of feedback. Proactive feedback depends on the state of the current solution, its criticality, and time factor based on when the last step has been carried out (Fossati,

Di Eugenio, Ohlsson, Brown, & Chen, 2010). Proactive feedback can be either negative or positive.

3. Methods and Data

We used log data from iList, version 5, as described in (Fossati et al., 2010). In our experiments, 32 college students worked with iList for approximately one hour long session during their introductory computer science data structures course. Students solved problems presented by the system at their own pace. In this study, we considered the first 5 problems, which use the interactive step-by-step mode. The system logged various actions taken by the student, including problem selection, programming commands, and undo/redo operations. Additionally, iList logged the automatically generated feedback provided to the students, and the metrics the system used to generate such feedback. In particular, iList recorded its probabilistic estimates of goodness "G" and criticality "C" of each solution step. These measures are crucial because most of the pedagogical decisions of iList are based on them. As we will see in section 4.3, our new logistic regression model can improve on the usage of these metrics alone.

We have extracted a number of features from the log data that will be used in the analysis of students' performance and interaction behavior. Some of these features have been used in previous research (Beck & Woolf, 2000; Mostow & Beck, 2006). Below is a description of these features:

The goodness (G) of state *S* – Explained in section 2

The criticality (C) of state *S* – Explained in section 2

Step duration – The time taken by the student to complete the step (milliseconds)

Relative step start time (RSST) – The start time of the step relative to the start time of the problem (milliseconds), it is the elapsed time since the start of the problem

Log step duration – The natural log transform of step duration

Log RSST – The natural log transform of relative step start time

Number of undos – Number of undos made so far in a given problem

Number of redos – Number of redos made so far in a given problem

Number of bad steps – Number of incorrect steps made in a given problem

Number of good steps – Number of correct steps made in a given problem

Syntax feedback– Number of syntax feedback messages so far in a given problem

Execution feedback– Number of execution feedback messages so far in a given problem

Negative reactive feedback – Number of negative reactive feedback messages

Positive reactive feedback – Number of positive reactive feedback messages

Negative proactive feedback – Number of negative proactive feedback messages

Positive proactive feedback – Number of positive proactive feedback messages

Step type – Refers to the type of actions performed. These can be: *Node Declaration*: creating a reference to a node, e.g. Node *T. *Node Creation*: creating a new node, e.g. T = new Node. *Delete Node*: deleting a node from a linked list, e.g. delete T. *Data Assignment*: assigning a value to the data field of a node, e.g. T.Data = 3. *Link Assignment*: assigning the link field of a node to either another link or null value, e.g. T.link = T1.Link, or T = T1, or T.link = null.

Target Step: The class label (1 = Correct, 0 = Incorrect), it represents the correctness of the step. It is determined based on the sign of the difference between the goodness G of the previous and current states. If the sign is positive then the step is considered correct, otherwise it is considered incorrect.

4. Experiments

We first start by exploring problem solving behavior relative to time. We used one-way ANOVA to find any significant temporal differences between step types, and between correct and incorrect steps. We then used logistic regression to build a predictive model of the correctness of a step, which can then be used to judge student performance. We evaluated a number of models for this purpose with different feature subsets. We used the SPSS Statistcs17 software package for the ANOVA and logistic regression analysis. Mathematical details of logistic regression can be found in (Hosmer, 2000).

4.1 Analysis of Students' Temporal Problem Solving Behavior

We were interested in exploring the problem solving behavior of students with respect to time (Step Duration). For this we conducted a number of ANOVAs where time is the response variable. The independent variables were the Step Type with five levels (Node Declaration, Node Creation, Delete Node, Data Assignment, Link Assignment), and Target Step. We used natural log transform of Step Duration because the original variable did not have a normal distribution. The ANOVA was significant for Target Step * log (Step Duration) $F(1, 1921) = 11.19$, $p < 0.05$: this indicates that students are taking more time to perform correct steps (with a mean M = 3.04), as compared to students who fall into errors or do incorrect steps, and who take less time (with a mean M = 2.90). ANOVA also showed a significant effect for Step Type * log (Step Duration), $F(4, 1918) = 18.50$, $p < 0.05$. Bonferroni posthoc tests revealed that Node Declaration (M = 3.25) and Data Assignment (M = 3.53) took more time and their means were significantly higher than the other step types, Link Assignment (M = 2.94), Node Creation (M = 2.77), and Node Deletion (M = 2.65). Additionally, Link Assignment was significantly different from Node Deletion but not from Node Creation. However this should be viewed within the context of our experimental protocol and the type of problems (exercises) described earlier. In order to shed some light on the relationship between Step Types and Target Step, we performed a chi-square independence test. The test was significant $X^2(4) = 211.13$, $p < .05$, which indicates that there is some type of dependency or relationship between step type and getting the step correct or not. Based on these results, we further explore this kind of relationships between variables via logistic regression analysis.

4.2 Predicting Students' Performance Using Logistic Regression

We choose logistic regression because we have a binary response variable (Target Step). We build models that are able to predict if the next step in a problem solving exercise will be correct or not. We choose to include all the variables (except Step Duration and Relative Step Start Time for which we include their log transform instead) in the regression model in order to check the prediction power of each individual variable. We include the Step Type in the regression model. One advantage of logistic regression is that it can handle categorical variables.

Table 1 shows the results of logistic regression; significant variables at the 0.05 are indicated by an asterisk next to the coefficient column. The Wald chi-square value and the odds ratios are also shown. We evaluate and compare two models for this task (Model 1 and Model 2). Model 1 incorporates all features listed in the column labeled Variable in Table 1, while Model 2 excludes the two probabilistic measures (G and C) which originally were used in iList for task modeling and feedback generation. Model 1 was significant with X^2 (18) = 1510.74 and $p < .05$, which suggests that this model with these independent variables included is more effective than the null model with intercept only (baseline model without any independent variables). The Hosmer-Lemeshow (H-L) goodness-of-fit test of the model was insignificant with X^2 (8) = 14.45 and p = .07, suggesting that the model fits the data well. Table 2 shows classification results of Model 1, it classifies 87.7% of instances correctly. It also shows the recall, and precision for each outcome. The recall for predicting the correctness of the step is 89.4%, which is slightly higher than that of incorrect steps of 86%. However the precision is higher for the latter with 88.8%, and 86.6% for the former. On the other hand, Model 2 was also significant with X^2 (16) = 964.95 and $p < .05$, however the H-L test was significant with X^2 (8) = 153.51 and $p < .05$, which indicates that the model is not quite good fit for the data. Model 2 classifies 81.6% of instances correctly. We also evaluated a logistic regression model with only G, C, and Log Step Duration; the model was able to classify only 80.4% of instances correctly.

From Model 1 we can see that the most predictive variables are: G, Number of Bad Steps, C, Log Step Duration, Step Type, Number of Good Steps, Number of Undos, Number of Redos, and Feedback Negative Reactive. Coefficients that are close to zero or odds ratios that are equal to one are not useful predictors. Furthermore, for all continuous variables in the model, the coefficient represents the change in log odds for each one unit increase of that independent variable. The model suggests a number of observations. First, chances that students will make correct steps increase if they spend more

Table 1: Logistic regression results, predicted variable Target Step = 1

	Model 1			Model 2		
Variable	Coefficient	Wald	Odds	Coefficient	Wald	Odds
Step Type		80.00			99.15	
Step Type (1)- Data Assignment	.43	.94	1.53	.72	3.69	2.06
Step Type (2)- Delete Node	.37	.73	1.45	.52	2.62	1.69
Step Type (3)- Link Assignment	.06	.05	1.06	1.03*	17.40	2.80
Step Type (4)- Node Declaration	2.67*	47.85	14.43	3.04*	85.82	20.98
Log RSST	-.27*	3.87	.76	-.23*	4.17	.79
Log Step Duration	.54*	30.62	1.72	.17*	5.01	1.19
Number Of Good Steps	-.37*	14.80	.69	-.05*	.53	.95
Number Of Bad Steps	-1.37*	147.54	.26	-1.24*	177.89	.29
Number Of Redos	-.78*	85.48	.46	-.62*	80.88	.54
Number Of Undos	.56*	107.23	1.75	.42*	90.84	1.52
Feedback Syntax	.02	.08	1.02	-.01	.06	.99
Feedback Execution	.07	.63	1.08	.02	.07	1.02
Feedback Positive Final	.06	.23	1.06	.21*	4.84	1.23
Feedback Negative Reactive	.17*	6.46	1.19	.28*	25.22	1.32
Feedback Positive Reactive	-.02	.16	.98	.21*	18.58	1.23
Feedback Positive Proactive	.02	.05	1.02	-.18*	11.90	.83
Goodness (G)	4.15*	261.23	63.33			
Criticality (C)	-1.10*	22.19	.33			
Constant	-.80	2.03	.45	-.26	.31	.77
Model Chi-Square [df]	1510.74 [18]			964.95 [16]		
H-L Chi-Square [df]	14.45 [8]			153.51 [8]		
% Correct Predictions	87.7			81.6		

* Statistically significant, P < .05

time on them, as indicated by the sign of coefficient of Log Step Duration. Similarly, the chances of getting the next step correct decrease relative to the number of bad steps in a given exercise. Likewise, it appears that the redo's and undo's work in opposite fashion as indicated by the sign of their coefficients, where the chances of getting a correct answer on the next step decreases with the number of redo's performed and increases with the number of undo's. The G and C scores which encode probabilistic measures of getting a correct or wrong answer also work in an opposite fashion as indicated from the signs of their coefficients. Most of the feedback types were not significant in this model other than Negative-Reactive feedback which has a positive coefficient, indicating the chances of performing a next correct step increase with more feedback of this type. The model also shows that Step Type is a significant predictor, however some of coded variables (Step Type(1) Data Assignment, Step Type(2) Delete Node, Step Type(3) Link Assignment) are not, where the reference baseline category was set to the first (Create Node) for the purpose of logistics regression. The odds of getting a Node Declaration correct are almost 14 times higher than the reference category. The ANOVA analysis we presented earlier revealed that Node Declaration took more time from students. This might explain the relationship between time spent on a step and getting it correct.

In Model 2, the list of significant features included the proactive and reactive feedback features. While both positive and negative reactive feedbacks have positive coefficients, the positive proactive feedback has a negative coefficient. This suggests that these types of feedback have an opposite effect on student performance. It is also noted that the feature Log RSST has a negative coefficient in both models. This indicates that some students are relatively slow in solving a problem, and this might raise the chances of producing errors. However, this effect is stronger in Model 2 compared to Model 1.

Table 2: Classification results of Model 1, observed and predicted frequencies for predicting step correctness by logistic regression, baseline 50.4% for Target Step = 1.

		Predicted			
		Target Step			
Observed		0	1	% Correct (Recall)	% Precision
Target Step	0	819	133	86	88.8
	1	103	866	89.4	86.6
	Overall % Correct			87.7	

4.3 Discussions and Conclusions

Our analysis showed that time has an impact on students' ability to perform correct or incorrect steps. The more time students spent on a particular step the higher chances they will perform it correctly, and vice versa. This also indicates that there are two types of interaction behaviors. First, students who put more thought into a problem, and students who jump straight into solving a problem without giving proper thought. While the former behavior should be encouraged, the latter should not. Proper interventions can be designed to mediate both types of interactions. For example, the system might not allow the student to type an answer straight, and enforce some time gap between the problem presentation and when the student can enter his/her answer. However, this could be done relative to the student historical performance, in order to differentiate between high and low performing students.

In our earlier work, (Fossati et al 2010), the two probabilistic measures G and C (goodness and criticality of a state), and time, were the only variables used to predict students' performance. We have seen an increase by almost 7% in the prediction power for students' performance after adding the set of derived features which were obtained from the log files. Using the set of derived features alone has almost the same prediction power as the probabilistic measures G and C. However, not all these features were significant predictors as indicated by the results of logistic regression of Model 2 presented in Table 1. It is also apparent from the regression results that there is tendency of falling into errors as more steps are performed, as indicated by the effect of Number Of Bad Steps predictor, and to some extent by the Number Of Good Steps.

We were able to achieve acceptable results of predicting students' performance on a particular task. The classification models built in this study can be used for predicting and tracking students' performance and providing proper interventions and guidance through the learning process. The results from this study will also enforce some design decisions (e.g. interface design, computational models) for future versions of iList; including our new ITS "ChiQat". The ChiQat project aims at building an intelligent tutoring system for teaching basic data structures including lists, trees, stacks, and recursion.

Acknowledgements

This work was supported by award NPRP 5-939-1-115 from the Qatar National Research Fund.

References

Baker, R., & Yacef, K. (2009). The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining, 1*(1), 3-17.

Beck, J., & Woolf, B. P. (2000). *High-Level Student Modeling with Machine Learning*. Paper presented at the Proceedings of the 5th International Conference on Intelligent Tutoring Systems.

Cen, H., Koedinger, K., & Junker, B. (2006). Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement. In M. Ikeda, K. Ashley & T.-W. Chan (Eds.), *Intelligent Tutoring Systems* (Vol. 4053, pp. 164-175): Springer Berlin Heidelberg.

Cetintas, S., Si, L., Xin, Y. P., & Hord, C. (2010). Predicting Correctness of Problem Solving in ITS with a Temporal Collaborative Filtering Approach. In V. Aleven, J. Kay & J. Mostow (Eds.), *Intelligent Tutoring Systems* (Vol. 6094, pp. 15-24): Springer Berlin Heidelberg.

Fossati, D., Di Eugenio, B., Brown, C., Ohlsson, S., Cosejo, D. G., & Chen, L. (2009). Supporting Computer Science Curriculum: Exploring and Learning Linked Lists with iList. *IEEE Transactions on Learning Technologies, 2*(2), 107-120.

Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, C., & Chen, L. (2010). Generating Proactive Feedback to Help Students Stay on Track. In V. Aleven, J. Kay & J. Mostow (Eds.), *Intelligent Tutoring Systems* (Vol. 6095, pp. 315-317): Springer Berlin Heidelberg.

Hosmer, D. W. L. S. (2000). *Applied logistic regression*. Chichester: Wiley.

Mostow, J., & Beck, J. (2006). Some useful tactics to modify, map and mine data from intelligent tutors. *Natural Language Engineering, 12*(2), 195-208.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review, 103*, 241-262.

Ohlsson, S. (2008). Computational models of skill acquisition. In R. Sun (Ed.), *The Cambridge handbook of computational psychology* (pp. 359-395). Cambridge, UK: Cambridge University Press.

Romero, r., Ventura, S., Espejo, P. G., & Hervás, C. (2008). *Data mining algorithms to classify students.* Paper presented at the Proc. of the 1st Int. Conf. on Educational Data Mining (EDM'08).