Predicting student help-request behavior in an intelligent tutor for reading

Joseph E. Beck, Peng Jia, June Sison, and Jack Mostow

joseph.beck@cmu.edu
http://www.cs.cmu.edu/~listen
Project LISTEN
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213. USA.

Abstract. This paper describes our efforts at constructing a fine-grained student model in Project LISTEN's intelligent tutor for reading. Reading is different from most domains that have been studied in the intelligent tutoring community, and presents unique challenges. Constructing a model of the user from voice input and mouse clicks is difficult, as is constructing a model when there is not a well-defined domain model. We use a database describing student interactions with our tutor to train a classifier that predicts whether students will click on a particular word for help with 83.2% accuracy. We have augmented the classifier with features describing properties of the word's individual graphemes, and discuss how such knowledge can be used to assess student skills that cannot be directly measured.

1 Introduction and Motivation

Project LISTEN's Reading Tutor [10] is an intelligent tutor that listens to students read aloud with the goal of helping them learn how to read English. Target users are students in first through fourth grades (approximately 6- through 9-year olds). Students take turns with the Reading Tutor picking stories to read. Students are shown one sentence at a time, and the Reading Tutor uses speech recognition technology to (try to) determine which words the student has read incorrectly [13]. The student can also request help on words about which he is uncertain; types of help include sounding out the word, pronouncing the word, and providing rhyming hints.

The Reading Tutor is more effective at helping children learn to read than simply letting children read books on their own [11]. Much of the Reading Tutor's power comes from allowing children to request help and from detecting some mistakes that students make while reading. It does not have the strong reasoning about the user that distinguishes a classic intelligent tutoring system, although it does base some decisions, such as picking a story at an appropriate level of challenge, on the student's reading proficiency.

Our goal is to strengthen the Reading Tutor's user model and determine which user characteristics are predictive of student behavior. If we can predict how students

behave while using the Reading Tutor, we can use such knowledge to help adapt the tutor's behavior, assess students, and we will have discovered useful features for understanding students using the tutor. Given our current lack of knowledge of good features for student modeling within the Reading Tutor, learning how to describe students is a worthwhile goal. We take the approach of observing users and trying to learn patterns from their behavior. Being able to validate hypotheses about how users behave against already logged data also simplifies experimental design: it is not necessary to run a new study to evaluate the accuracy of the student model.

2 Approach

In this section we discuss how we collected data from students, what aspect of the students we choose to model, how to approach the problem as a classification task, and various architectures for training the classifier.

2.1 Data Collected

In the 2000-2001 school year, 88 students in two schools used the Reading Tutor as part of a controlled study. Students used the Reading Tutor from the end of October 2000 until the beginning of June 2001. On average, students used the Reading Tutor for approximately 18 hours.

The Reading Tutor logged when students started reading a story, when a new sentence was displayed, when a student read each word in the sentence, and when the student clicked on a word for help. These data were parsed and used to construct an SQL database [12]. Across the 88 students, students saw a total of 2.4 million words over the year, which amounts to students seeing 25 words per minute while using the Reading Tutor (about one-tenth of the average adult reading rate).

Students requested help on a word from the Reading Tutor 229,000 times over the course of the year. We know what word the student clicked on and when he clicked.

Unfortunately, the logging was incomplete. We are not able to extract from the logs what type of help the Reading Tutor provided to the student. Since many students would continue to click until the Reading Tutor read the word (the Reading Tutor randomly selects which type of help to give each time the student clicks), it is difficult to interpret what multiple requests for help mean: the student could be very confused or simply holding out until the RT read the word.

Another area where logging was incomplete was when the Reading Tutor decided to provide help to the student. The Reading Tutor gives help to students before they start reading a sentence on words that it thinks are difficult. The rationale for this help is that if a student gets practice reading a word incorrectly it will be harder to teach him later the correct way of reading the word [4]. Therefore, the Reading Tutor gives help on words the student is likely to have difficulties with. Unfortunately, whether the Reading Tutor gave preemptive assistance was logged, but the actual word for which the tutor provided assistance was not recorded in an easily analyzable form.

2.2 Modeling Problem: What to Predict?

Predicting student behavior is a means of adapting instruction. If the tutor knows when the student will, for example, require assistance, it can provide help before the student becomes stuck. There are a variety of student behaviors we could predict: whether the student will speak a word correctly, the delay before a student begins to pronounce a word [6, 9], whether the student clicks on a word for help, etc. Speech recognition is a far from perfect technology, so to simplify our task we concentrate on predicting whether the student will click for help on a word.

We treat predicting whether the student asks for help as a binary prediction task. One use for such a model is to determine when the Reading Tutor should provide preemptive assistance before the student attempts to read the sentence. Currently, preemptive assistance is biased towards giving help on longer words and on words the student has read poorly in the past. This approach is somewhat lacking as it does not consider the length of time since the student saw the word, whether the student has previously asked for help on this word, etc. A model of help requests is also a model of relative word difficulty: words that a student is likely to click for help on are harder than words for which the student does not need help. Such a model of word difficulty could be used to select stories that are appropriate for the student, or to provide additional information to speech recognizer. In particular, if a student is likely to need help on a word (and does not receive help) the prior probability that he will read the word correctly should be lowered.

We examined student help request patterns on common and uncommon words. Dolch words [3] are 220 high-frequency words that children should learn by sight. Students tend to have few help requests on Dolch words, so we do not consider them in our analysis. Figure 1 shows student help request patterns on non-Dolch words. The x-axis is how many times the student has seen this word previously in the Reading Tutor. The y-axis is the probability he will request help on a word. Each line on the curve represents a group of students who scored similarly on the Woodcock Reading Mastery's Word Identification test [15]. The Word Identification (WI) test reports student's ability to read words correctly in English. A score of 2.3 means that a student reads at a level comparable to a student in the 3rd month of the 2nd grade. Students were pre-tested in October before using the Reading Tutor and post-tested in late March. We define a student's *grade level* as the average of pre- and post-test scores on the WI test. This figure excludes students who had grade levels of 4, 5, or 6 for clarity (their help request rate was lower than students at grade level 3).

Figure 1 shows a strong effect for students' reading ability; students at higher grade level ask for help less often. Students initially ask for help on words fairly often, but after being exposed to words ask for help less often. For example, someone with an average Grade level of 1 asks for help about 20% of the time on a word he has seen only once. After seeing a word 10 times, students ask for help only ≈6% of the time on average. We have marked the region of Figure 1 that we have chosen to try to model. We are focusing on an "interesting" region of the space. For students at higher grade levels, it is easy to predict accurately whether the student will ask for help: simply predicting "no" does an excellent job. Since one use of our model of student help requests is to decide when to give preemptive assistance, focusing on

students who need assistance is a good first step. Therefore, we concentrate on students whose grade level is 1 or 2.

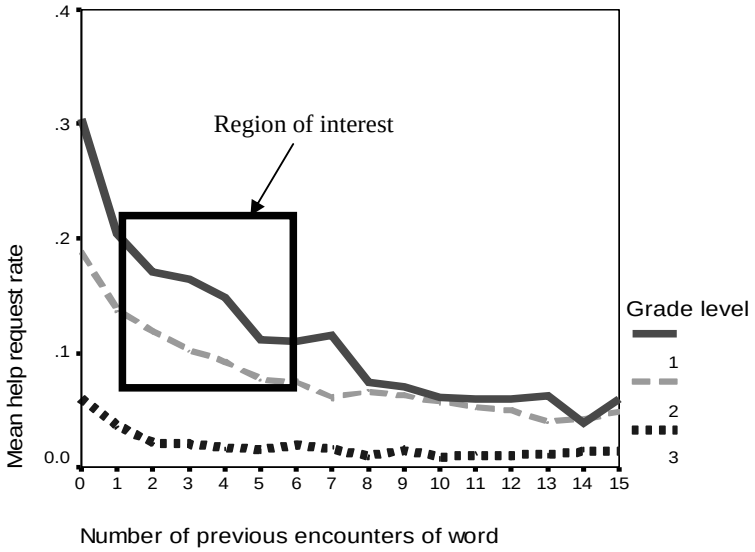

Figure 1. Learning curves showing students decreasing help request rate

The first time a student sees a word, preemptive assistance is almost always provided. Since we do not know whether the Reading Tutor gave assistance on a word, it is difficult to predict whether the student will click on the word (presumably if the Reading Tutor provides help the student will not also click for help). Therefore, we ignore first encounters of a word. After a student has seen a word 6 times the chances of asking for help become rather low. Therefore, we focus on words that have only been seen 6 or fewer times.

This screening procedure reduces our sample of students from 88 to 53, and the number of word encounters from 2.4 million to 555,818.

2.3 Casting the Problem as a Classification Task

Since our goal is to determine whether the student will click for help on a word, and we know which words the student clicked or did not click for help, we need to find some method of describing words that will allow us to predict into which category a particular word falls. We consider information that is static for a particular encounter of a word, such as the student's reading proficiency and past history with reading and asking for help on this word.

For each word, we computed **static properties** such as its length and its frequency of occurrence in children's stories. We also computed word properties that are not-static, but are **constant for a particular sentence** such as the word's position in the sentence. Features about position were encoded both as an absolute and as a

percentage of the length of the sentence (i.e. a word at position 3 in a 12 word sentence would be 0.25).

We used **properties about the student**, including gender and grade. We also use the student's current WI score for the day when he encountered each word. We estimate his WI score by assuming that it increases linearly between the pre- and post-tests. For a coarse measure on a grade equivalent scale, linear growth is a reasonable assumption. Linear growth is not a reasonable assumption for a more specialized skill such as solving a linear equation. For this research we used the linearly interpolated WI score provided by the paper tests; a deployed Reading Tutor would obviously not have access to the student's posttest score, so would have to use its estimates of the student's WI level [1, 6]. Another feature we used is the student's overall rate of clicking for help on words.

We also used the **student's history of reading this word** as a source of information for features. Information extracted from the log files includes how many days have passed since the student last encountered this word, the average latency [9] (delay before pronouncing a word) in past encounters, how many previous encounters the student has had with this word, and how many times the speech recognizer heard the student read this word.

The student's **help request behavior on this word** is also important. Features include whether the student has ever asked for help on this word, whether he has asked for help on this word today, and if the student asked for help the first time he saw this word.

We generated an exhaustive list of 60 features from the above sources. Knowledge of which features were likely to be redundant trimmed this list down to 20 features.

Our goal is to use these features to predict whether a student will ask for help on a particular word. We now turn to constructing a model to make such predictions.

2.4 Group vs. Individual Modeling: How to Construct Models

We use the term "group modeling" to refer to user models that are constructed from a group of users and applied to a new user, and the term "individual modeling" to refer to user models constructed with a user's own data. Some researchers use the terms "collaborative" and "feature-based" [16] respectively to describe such models. "Collaborative" is already an overused term in education circles, and "feature-based" is similarly misleading since many group models are feature-based.

Given that we have a series of labeled instances, features describing words and students, and information about which words the student clicked on for help, how can we construct a classifier to make predictions? We do not have sufficient training data to build a model for each word in the English language. Therefore, we collapse across words and rely on the features describing them (length and frequency) to describe meaningful differences.

We have a similar option for whether to build a separate classifier for each student [14] (individual modeling) or whether to aggregate the data for all of the students together to construct a group model [2, 5]. Advantages of aggregating the data together include more training instances than for any individual student. Having more data tends to result in more accurate models. Another advantage is that we can tune

the classifier's performance and once it is satisfactory, we can deploy it in the Reading Tutor.

Advantages of building a separate model for each student include possibly higher accuracy. If the features do poorly at describing students, or if some students differ dramatically, then constructing a model for each student might do a better job. We compare both group and separate models as means for constructing a classifier.

To evaluate the classifiers, we used a different penalty for different types of mistakes. A classifier can make two types of mistakes: it can predict that a student wishes to receive help when he doesn't, or it can predict that a student does not need help when in fact he does. We believe that the second mistake is more important to avoid. If a student needs help, he might not realize it and proceed to read the word incorrectly (young children often do not have strong meta-cognitive skills [8]). The Reading Tutor does not do a good job at detecting misread words. Also, due to occasionally mishearing correctly read words, the Reading Tutor permits the student to advance to the next sentence after reading at least 50% of the words in the sentence correctly. Therefore, it is possible for a student to complete a sentence without knowing how to read a particular word. If a student gets practice at incorrectly reading the word, then correcting him later will be more difficult. Table 1 shows how we penalize the classifier for various types of mistakes.

Table 1. Penalty matrix for classifier

	Classifier predicts help requested	**Classifier predicts no help requested**
Student clicks for help	0	5
Student does not click for help	1	0

3 Results

We now compare the results of grouped and individual modeling, and then discuss adding phonemic features to the classifier.

3.1 First Experiment: Group vs. Individual Modeling

To evaluate the performance of a group modeling agent, it is necessary to aggregate the data from multiple students to train the classifier. To test the classifier, we planned to evaluate its accuracy using a leave-one-out approach by constructing the classifier with the data from N-1 students, testing on the Nth student, and repeating the process for each student. However, we built the classifier with data from only 25% of the other students since using more of the data crashed the computer.

For individual modeling, we are trying to determine how our classifier performs if it were trained on the data from an individual student. One approach is to load **all** of he student's data, and perform a cross-validation to determine the model's accuracy. Such an approach is not appropriate since the data are temporally dependent on each other. For example, imagine the third time a student encounters the word "telephone"

is in the training set, and we know that he asked for help on both prior encounters. If the second time the student saw the word "telephone" is in the test set we can predict with perfect accuracy that he must have asked for help. Therefore, we incrementally add instances to the training data.

Figure 2 shows our approach for training an individual's model. The student's data are first sorted by time, and then the first item is presented to the classifier. The classifier makes a prediction for this one item, which is then added to the training data. This iterative process continues, such that over time the classifier's training data grows and is used to make a prediction for the next word encountered by the student.

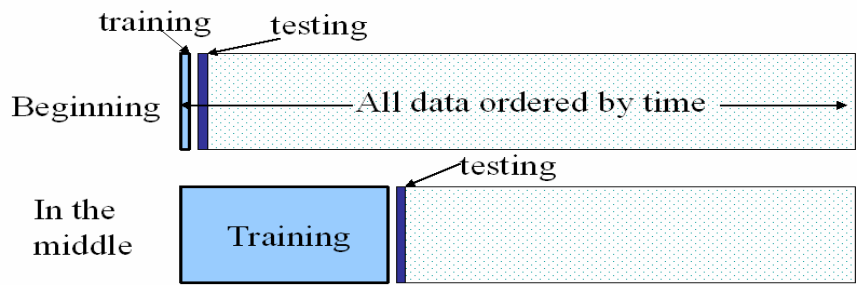

Figure 2. Individual modeling approach

We used Weka, a public domain data mining package written in Java, to perform these analyses. Specifically, we used Weka's J48 (a version of C4.5) and naïve Bayesian classifier (NBC) methods for performing the classifications.

For group modeling, J48 was correct on 71% of its predictions and NBC was correct on 75% of its predictions. For individual modeling, 81% of J48's predictions were correct and 75% of NBC's predictions were correct.

These results are somewhat surprising. Naïve Bayesian classifiers tend to outperform decision trees when there is a small amount of training data [7]. However, we found that decision trees work better for predicting on the basis of data from one student, while NBCs work better with data from multiple students.

Overall, individual modeling with J48 gave the best result, with an accuracy of 81%. One way to improve on this result is to use the model trained from a group of students until enough data are collected to construct a model from the current student's data. Another approach is to combine data from the current student with the data from the population to improve the model's predictions [2, 5].

3.2 Second Experiment: Adding Phonemic Information

Another goal of student modeling in the Reading Tutor is to assess "hidden skills." That is, skills that are not directly measured but that must be performed to solve a task. For reading, an example is the grapheme *ph* makes the sound /f/ in the word "phone." There are many letter to sound mappings in English, and testing the student on each of them would disrupt the natural process of reading.

As a first step towards this goal, we provide our classifier with information about the graphemes (letters and groups of letters) and grapheme to phoneme (the sound that

a grapheme makes) mappings that make up each word. A grapheme→phoneme mapping (abbreviated as "g→p mapping") is simply the grapheme and phoneme for a particular word paired together. E.g. the word date contains *D*→/D/, *A*→/EY/, and *TE*→/T/ as g→p mappings.

If such features are useful in helping to predict on which words students request help, then we can reason backwards and infer that students are having difficulties reading words with those particular hidden skills. This approach of adding information about the components that make up a word generalizes to other predictive tasks. For example, if we are predicting whether a student will read a word correctly, determining whether students make mistakes on words that contain rare graphemes will aid in drawing conclusions about a student's ability on these hidden skills.

We added phonemic features to the already existing features. Specifically, we added features about how likely various grapheme→phoneme mappings are in the English language. P(g→p) is defined as the word frequency weighted probability of a word containing this g→p mapping. Specific features were:

1. Average probability of all g→p mappings in the word
2. Probability of the first g→p mapping in the word
3. Probability of the rarest g→p mapping in the word

The first feature describes the overall unusualness in the word's pronunciation. The second feature accounts for the fact that students often try to read the first portion of a word and guess at the remainder of the word. The last feature models the most difficult part of the word. If the word is hard overall (feature #1), or if the student cannot get started in reading the word (feature #2), or if the student encounters a very rare mapping (feature #3), then we expect the student is more likely to ask for help.

We also added features about how likely it is for a grapheme to make a particular sound. That is, what is P(p|g). If a word contains a rare g→p mapping, such as in *CH* making the sound /k/ as in "chord," then we could expect students to have difficulties reading this word. For each word, we added the following features:

1. Average of P(p|g) for all graphemes in the word
2. P(p|g) for the first grapheme in the word
3. P(p|g) of the rarest grapheme in the word

Unfortunately, we did not have g→p mappings for all the words in our dataset. We had g→p mappings for only 2,460 out of 4,169 distinct words.

To run experiments, we first screened out words for which we did not have g→p mappings. Since constructing models for individual students outperformed building models from groups of students, we constructed a model for each student using J48. On the subset of words for which we had g→p mappings, our classifier had an accuracy of 82.5%.

Adding the six features described above gave an average accuracy of 83.2%. This marginal gain was statistically reliable at P=0.013 (paired samples t-test). Improvements in performance were not distributed randomly across students. There was a correlation of -0.46 between the classifier's performance with the original set of features and gain from adding phonemic information. I.e. for students who were already well-modeled, there was little gain from additional information.

We have demonstrated that information about the phonemic properties of words can help predict whether students will click for help on a word. This result suggests that a word's phonemic information affects whether students believe they know a

word and ask for help. Therefore, help request patterns are an avenue for assessing these hidden skills. If students request help on a word containing a set of g→p mappings, we can update the tutor's estimate of the student's proficiency on those skills. Fine-grained assessment of a student's knowledge is helpful for predicting what types of mistakes he is likely to make. For example, if a student can pronounce "chord," it is probable that he can also pronounce "chords," and will have a better chance at pronouncing "chaos" than a student who cannot read the word "chord." The common factor in all of these words is the initial grapheme /CH/ making the sound "k." Being able to assess student knowledge at the grapheme level would be a large improvement in the Reading Tutor's capabilities.

4. Conclusions and Future Work

Logging student interactions with the tutor in an easily analyzable form is a powerful strategy for constructing user models. We have used a database constructed from logged information to construct a student model capable of predicting whether the learner will click on a word for help. Such user models can be used to adapt the tutor's behavior by, for example, determining when to give the student help preemptively. Analyzing predictive user models can also be a means for assessing a student's proficiency at "hidden skills."

Our first attempt at adding phonemic features met with mixed success. Although the improvements in performance were significant statistically, they may not be meaningful. Finding a stronger set of features would benefit both predicting student help requests and relating a student's actions to his underlying knowledge of English words. One possibility for better features is to consider the student's history for each g→p mapping. E.g. what percentage of words containing this g→p mapping does the student read correctly? What percentage of words containing this g→p mapping does the student ask for help on? Such historical information was useful at the word level for predicting a student's help requests, so it is likely historical information at the grapheme level will also have predictive power.

One possibility is not all students are decoding words, but are instead recognizing them as whole words (or guessing). One way to determine which strategy students are using is to use the inter-word latency [9]. Presumably a decoding approach will result in a larger latency. We are exploring such approaches.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant No. REC-9979894. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation or the official policies, either expressed or implied, of the sponsors or of the United States Government. We thank members of Project LISTEN who contributed to this work, especially Susan Rossbach for conducting the field studies, and Andrew Cuneo for constructing the database from the logfiles.

References (see www.cs.cmu.edu/~listen for LISTEN publications)

1. Beck, J.E., Jia, P. and Mostow, J., Assessing Student Proficiency in a Reading Tutor that Listens. Proceedings of the *Ninth International Conference on User Modeling*. 2003
2. Beck, J.E. and Woolf, B.P., Using a Learning Agent with a Student Model. Proceedings of the *Fourth International Conference on Intelligent Tutoring Systems*. p. 6-15. 1998
3. Dolch, E., A basic sight vocabulary. *Elementary School Journal*, 1936. **36**: p. 456-460.
4. Hebb, D.O., *The Organization of Behavior*. 1949, New York: Wiley.
5. Jameson, A. and Wittig, F., Leveraging Data About Users in General in the Learning of Individual User Models. Proceedings of the *Seventeenth International Joint Conference on Artificial Intelligence*. p. 1185-1192. 2001
6. Jia, P., Beck, J.E. and Mostow, J., Can a Reading Tutor that Listens use Inter-word Latency to (better) Assess a Student's Reading Ability? Proceedings of the *ITS 2002 Workshop on Creating Valid Diagnostic Assessments*. 2002
7. Langley, P., Iba, W. and Thompson, K., An Analysis of Bayesian Classifiers. Proceedings of the *National Conference on Artificial Intelligence*. p. 223-228. 1992
8. Lundberg, I. and Olofsson, A., Can computer speech support reading comprehension? *Computers in Human Behaviour*, 1993. **9**(2-3): p. 283-293.
9. Mostow, J. and Aist, G., The Sounds of Silence: Towards Automated Evaluation of Student Learning in a Reading Tutor that Listens. Proceedings of the *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. p. 355-361. 1997
10. Mostow, J. and Aist, G., Evaluating tutors that listen: An overview of Project LISTEN, in *Smart Machines in Education*, K. Forbus and P. Feltovich, Editors. 2001. p. 169-234.
11. Mostow, J., Aist, G., Bey, J., Burkhead, P., Cuneo, A., Junker, B., Rossbach, S., Tobin, B., Valeri, J. and Wilson, S., Independent practice versus computer-guided oral reading: Equal-time comparison of sustained silent reading to an automated reading tutor that listens. Proceedings of the *Ninth Annual Meeting of the Society for the Scientific Study of Reading*. 2002
12. Mostow, J., Beck, J., Chalasani, R., Cuneo, A. and Jia, P., Viewing and Analyzing Multimodal Human-computer Tutorial Dialogue: A Database Approach. Proceedings of the *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces (ICMI 2002)*. p. 129-134. 2002
13. Mostow, J., Roth, S.F., Hauptmann, A.G. and Kane, M., A prototype reading coach that listens [AAAI-94 Outstanding Paper Award]. Proceedings of the *Proceedings of the Twelfth National Conference on Artificial Intelligence*. p. 785-792. 1994
14. Webb, G.I. and Kuzmycz, M., Feature Based Modelling: A methodology for producing coherent, consistent, dynamically changing models of agents' competencies. *User Modeling and User Adapted Interaction*, 1996. **8**: p. 97-115.
15. Woodcock, R.W., *Woodcock Reading Mastery Tests - Revised (WRMT-R/NU)*. 1998, Circle Pines, Minnesota: American Guidance Service.
16. Zukerman, I. and Albrecht, D.W., Predictive Statistical Models for User Modeling. *User Modeling and User Adapted Interaction*, 2001. **11**(1-2): p. 5-18.