Does Help Help?
A Bayes Net Approach to Modeling Tutor Interventions

Kai-min Chang, Joseph E. Beck, Jack Mostow, and Albert Corbett

Project LISTEN, School of Computer Science
Carnegie Mellon University, Pittsburgh PA, 15213, USA
kkchang@cs.cmu.edu

Abstract

This paper describes an effort to measure the effectiveness of tutor help in an intelligent tutoring system. Conventional pre- and post- test experimental methods can determine whether help is effective but are expensive to conduct. Furthermore, a pre and post- test methodology ignores a source of information: students request help about words they do not know. Therefore, we propose a dynamic Bayes net (which we call the *help* model) that models tutor help and student knowledge in one coherent framework. The *help* model distinguishes two different effects of help: scaffolding immediate performance vs. teaching persistent knowledge that improves long term performance. We train the *help* model to fit the student performance data gathered from usage of Reading Tutor. The parameters of the trained model suggest that students benefit from both the scaffolding and teaching effects of help. Thus, our framework is able to distinguish two types of influence that help has on the student, and can determine whether help helps learning without an explicit controlled study.

Introduction

An important property of an Intelligent Tutoring System (ITS) is its ability to help students. Thus, a basic question in ITS is to measure the effectiveness of its help. Does help help? Does one type of help work better (Heiner, Beck et al. 2004)? Even though the tentative answer is *yes* by most ITS researchers, (otherwise, why include help at all in the tutor?), answering such questions is surprisingly difficult. Furthermore, the question of "does help help?" is ill-defined; what does it mean to help students? This paper describes an effort to specify how help impacts student and measure the effectiveness of an ITS's help.

Ideally, we would like to set up a controlled pre- and post-test experiment to measure the effectiveness of tutor help. A typical experimental setup works as follows: in the pre-test, we assess student performance before using the ITS. Then, we will randomly assign students into two groups. The experimental group uses one version of ITS *with* the tutor help that we're evaluating, whereas the control group uses another version of ITS *without* the particular tutor help. After students use the ITS for some time, we assess student performance of the two groups again in the post-test. Finally, we test the hypothesis that the performance improvement in the experimental group is significantly different than the control group. Although this experimental design is sound and reliable, a controlled experiment takes a long time to conduct and is often too expensive to conduct although exceptions exist (e.g. Arroyo, Beck et al. 2001).

Given that the ideal pre- and post-test experimental studies are often impractical, there are several other approaches to measure the effectiveness of tutor help. For example, we may want to conduct user case studies and directly ask the students whether they find the tutor help effective. Unfortunately, while user case studies provide valuable qualitative feedback, they lack the ability to draw conclusive causal-relationships. Alternatively, we can try to infer tutor help efficacy *from the data*. For instance, one might claim that a tutor help is effective if student performance improves when they receive help, compared to when they do not receive help. However, this approach raises the question of *when* to assess student performance. Immediate performance is prone to scaffolding effects where tutor help merely provides a short-term performance boost. For example, some help types provide students the answer; if students simply mimic the help, should we count that as learning?

To measure the effect of help on persistent knowledge learning, we can use delayed performance. Since what we care about is not just the effect of tutor help on student performance, but rather its effect on student knowledge. In an ITS, student knowledge is often modeled by a student model that describes the learner's proficiencies at various skills. Indeed, tutor help and student model are two tightly coupled components in an ITS. For example, tutor help is mostly provided where the student has the least knowledge. Despite the close relationship between student model and tutor help, the two components are often evaluated and modeled independently (e.g. Heiner, Beck et al. 2004; Beck, Jia et al. 2004).

Compilation copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

In this paper, we describe a methodology to model both tutor help and the student's knowledge in one coherent framework. This configuration allows us to tease apart the effect of help into 1) scaffolding immediate performance and 2) teaching persistent knowledge that improves long term performance. First, we first describe prior work on assessing student knowledge using Knowledge Tracing and dynamic Bayes nets. Then, we demonstrate how dynamic Bayes nets can be used to simultaneously model the effectiveness of tutor help and student knowledge. We evaluate the proposed framework with student performance data in Reading Tutor. Finally, we conclude with contributions and future work.

Prior Work on Assessing Student Knowledge

Knowledge Tracing

Knowledge Tracing (KT) (Corbett and Anderson 1995) is an established technique for student modeling and was first used in the ACT Programming Languages Tutor. The goal of knowledge tracing is to estimate student's knowledge from their observed actions. Prior work in this area (Beck and Sison 2004) has shown that KT is an effective approach for ITS that use ASR output to model students.

As illustrated in Figure 1 and Equation 1, KT maintains four constant parameters for each skill. Notice, KT assumes there is no forgetting, so the *forget* parameter is set to 0. Two parameters, *already know* and *learn*, are called learning parameters and refer to the student's initial knowledge and to the probability of learning a skill given an opportunity to apply it, respectively. Two other parameters, *guess* and *slip*, are called performance parameters and account for student performance not being a perfect reflection of his underlying knowledge. The guess parameter is the probability that a student who has not mastered the skill can generate a correct response. The slip parameter is used to account for even knowledgeable students making an occasional mistake.

At each successive opportunity to apply a skill, KT updates its estimated probability that the student knows the skill, based on the skill-specific learning and performance parameters and the observed student performance (evidence).

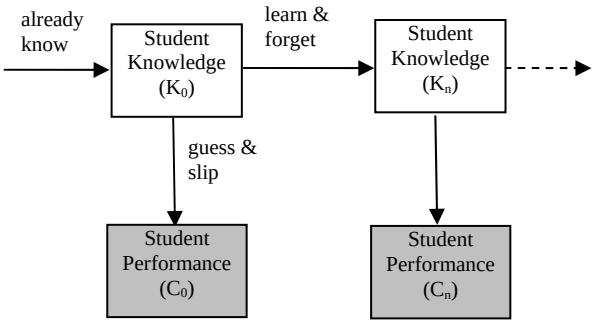

Figure 1 Graphical representation of Knowledge Tracing

$$\begin{aligned} already\ know &:= \Pr(K_0 = true) \\ learn &:= \Pr(K_n = true \mid K_{n-1} = false) \\ forget &:= \Pr(K_n = false \mid K_{n-1} = true) = 0 \\ guess &:= \Pr(C_n = true \mid K_n = false) \\ slip &:= \Pr(C_n = false \mid K_n = true) \end{aligned}$$

Equation 1 Parameters of Knowledge Tracing

Dynamic Bayes Net

Dynamic Bayes Nets (DBNs; Dean 1989) are another technique that have been applied to model student knowledge in ITS (Conati, Gertner et al. 2002). Reye (Reye 2004) showed that KT is special case of a DBN.

In previous research, we implemented a generic Bayes net toolkit (BNT-SM; Chang, Beck et al. 2006) for student modeling. BNT-SM inputs a data set and a compact XML specification of a DBN model hypothesized by a researcher to describe causal relationships among student knowledge and observed behavior. It generates and executes the code to train and test the model using the Bayes Net Toolbox (Murphy 1998). BNT-SM allows researchers to easily explore different hypothesis with respect to the knowledge representation in a student model. We show how to use BNT-SM to construct a DBN that models the effectiveness of tutor help on student knowledge in the next section.

Using DBN to Assess Effect of Help on Student Knowledge

Students often receive help on skills where they have the least knowledge. Despite the close relationship between student knowledge and tutor help, the two components are often modeled independently. In this section, we describe how to use DBNs to model both tutor help and the student's knowledge simultaneously.

The DBN that we propose here extends KT to include an additional *help* node, representing whether or not the student receives tutor help while performing a skill. We

call this network the *help* model. By modeling the tutor help and student knowledge simultaneously, the *help* model allows us to tease apart two different effects of tutor help: 1) scaffolding and 2) teaching. Whereas the effect of scaffolding is immediate (helping student to perform correctly in the current attempt), the effect of teaching is persistent (helping student to learn the knowledge and perform better on future problems). Figure 2 depicts the two effects of tutor help. Notice that Figure 2 is identical to Figure 1, with the addition of the new *help* node and the *teach* and *scaffold* links.

Equation 2 shows how the learning and performance parameters are computed in the *help* model. Notice how the KT model and the *help* model represent these parameters differently. For example, whereas the KT model models a single *learn* parameter through Pr(learn), the *help* model models 2 *learn* parameters through Pr(learn|help) and Pr(learn|no help). The *help* model conditions the parameters on whether or not tutor help is provided.

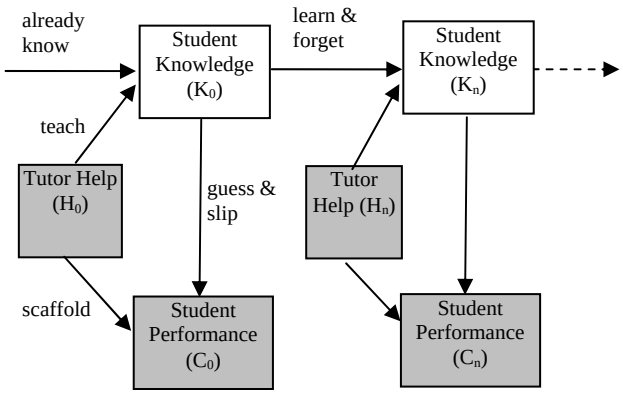

Figure 2 Graphical representation of the Help model

$$already\ know \mid help := \Pr(K_0 = true, \mid H_0 = true)$$
$$already\ know \mid no\ help := \Pr(K_0 = true, \mid H_0 = false)$$

$$learn \mid help := \Pr(K_n = true \mid K_{n-1} = false, H_n = true)$$
$$learn \mid no\ help := \Pr(K_n = true \mid K_{n-1} = false, H_n = false)$$

$$forget \mid help := \Pr(K_n = false \mid K_{n-1} = true, H_n = true)$$
$$forget \mid no\ help := \Pr(K_n = false \mid K_{n-1} = true, H_n = false)$$

$$guess \mid help := \Pr(C_n = true \mid K_n = false, H_n = true)$$
$$guess \mid no\ help := \Pr(C_n = true \mid K_n = false, H_n = false)$$

$$slip \mid help := \Pr(C_n = false \mid K_n = true, H_n = true)$$
$$slip \mid no\ help := \Pr(C_n = false \mid K_n = true, H_n = false)$$

Equation 2 Parameters of the Help model

Evaluate the Effect of Help

Given the proposed *help* model, we now train the model to fit the student performance data gathered from usage of Reading Tutor. The trained model parameters will allow us to evaluate the effectiveness of tutor help by seeing whether help impacts the learn parameters (teach) or the guess parameters (scaffold). Moreover, we also train the original KT model to compare how the two models fit student performance data.

Data Collection

Our data came from 360 children between six and eight years old who used Project LISTEN's Reading Tutor (Mostow and Aist 2001) in the 2002-2003 school year. Over the course of the school year, these students read approximately 1.95 million words (as heard by the automatic speech recognizer). On average, students used the tutor for 8.5 hours. During a session with the Reading Tutor, the tutor presented one sentence (or fragment) at a time for the student to read aloud. The student's speech was segmented into utterances delimited by silences. Each utterance was processed by the Automatic Speech Recognizer (ASR) and aligned against the sentence. This alignment scored each word of the sentence as either accepted (thought by the ASR to be read correctly) or rejected (thought to be misread or omitted). For modeling purposes, this paper treats each English word as a separate skill.

Parameter Estimation

We first separated the training and testing set by splitting the students into two groups. The split was done by sorting the students according to their amount of Reading Tutor usage and alternately assigning students to the two sets. We used the Expectation Maximization (EM) algorithm to optimize the data likelihood (i.e. the probability of observing our student performance data). EM is the standard algorithm used in the machine learning community to estimate DBN parameters when the structure is known and there exist latent variables. EM is guaranteed to converge to a local maximum on the likelihood surface. We used the junction tree algorithm for exact inference. We train two DBNs: one models the traditional KT model and the other one model the *help* model which models the effect of tutor intervention on student performance.

Evaluation of Student Model

Since student knowledge is a latent variable that cannot be directly observed, we have no gold standard to compare against. Instead, we used the trained student model to predict whether the ASR would accept or reject a student's next attempt of the problem. That is, we observe reading item by item and the task is to predict whether next word will be read correctly (in unseen test data). An ROC (Receiver Operating Characteristic) curve measures the

performance of a binary classifier by plotting the true positive rate against the false positive rate for varying decision thresholds. The area under the ROC curve (AUC) is a reasonable performance metric for classifier systems, assuming no knowledge of the true ratio of misclassification costs (Hand, Mannila et al. 2001).

To evaluate the accuracy of our model's internal estimate of student knowledge, we compare both the AUC and the correlation between its estimates and student performance (as scored by the ASR) on the held out test data.

As Table 1 shows, the simpler KT model outperforms the *help* model on both the correlation and the AUC evaluation metrics. Although the two correlation coefficients are reliably different at $p < 0.01$, their values are quite comparable. The values of model fit appear low because we are predicting individual student performance data rather than aggregated performance. It is difficult to predict a student's individual responses. We performed a cheating experiment to determine the best correlation with student performance that a student model could achieve. The cheating experiment allowed the student model to peak at future data before making a prediction. However, the cheating model was limited in that it had to make the same prediction for the current item as it made for the prior item, unless its last estimate was incorrect. Another way of thinking of this model is that it is monotonic. That is, if it predicts a student will get an item correct and he gets it correct, the model cannot then backtrack and predict he will get the next item incorrect. It must first receive evidence that it has over- or under-estimated the student's knowledge before changing its prediction. The cheating experiment revealed that the maximum correlation of any model that obeyed monotonicity constraints was only 0.5. Note that this maximum performance requires peeking at the data to be predicted and is not necessarily attainable by any actual model.

Table 1. Comparing the model fit of the KT model and the *help* model on held out test data.

	Correlation	AUC
KT	0.144	0.615
Help	0.135	0.612

Evaluation of the Effectiveness of Tutor Help

To evaluate the effectiveness of tutor help, we now compare the model parameters estimated by both the KT model and the *help* model.

Table 2 shows the parameters estimated for the KT model and the *help* model, respectively. Notice, the KT model does not consider the help information, whereas the *help* model models the parameters conditioned on whether or not tutor help is given or not. Thus, even though the *help* model has worse model fit to the KT model, the *help* model is more informative than the KT model. Notice, model informativeness is sometimes at odd with model fit. For example, structural equation models usually do not fit data as well as simple regression, but are more interpretable and lend themselves to create new hypothesis (Cohen 1995). In this case, the informativeness outweighs the slight decrease in model fit.

As seen in the Help model of Table 2, the probability of *already know* is much higher when there is no help than when there is help. This suggests that tutor helps are more likely to be provided to harder words – a positive finding. Also, the probability of *learning* is higher when there is help than when there is no help. Even though the difference is small, it is at least in the right direction, suggesting that tutor helps do have an effect on long term learning.

Table 2. Comparing the parameters estimated by the KT model and the *help* model.

	KT model	Help model	
		No Help Given	Help Given
Already Know	0.618	0.660	0.278
Learn	0.077	0.083	0.088
Guess	0.689	0.655	0.944
Slip	0.056	0.058	0.009

Also seen in the Help model in Table 2, the probability of guess is higher when there is help than when there is no help on the first encounter and that the probability of slip is higher when there is no help than when there is help. Both the learning and scaffolding effects are statistically reliable at $p<0.05$ (paired samples t-test, weighted by number of observations for each skill), suggesting that tutor help does have an effect on student performance.

Contribution

In this paper, we have proposed a methodology to infer the efficacy of help from observational data rather than experimental data. One question that we are interested to explore is how this framework compares to the pre- and post- test experimental design (Arroyo, Beck et al. 2001). Do they draw similar conclusions, despite the fact that an experimental design is usually more expensive to conduct than data fitting with DBNs? Moreover, what kind of causal relationship can we conclude with Bayes nets?

The second contribution this paper makes is on simultaneous representation of tutor help and the student model. Previous approaches addressed these problems separately by ignoring one to solve the other (Heiner, Beck et al. 2004; Beck, Jia et al. 2004). Specifically, KT ignored help, and most embedded experiments (Mostow and Aist 2001) ignored student knowledge, or how it changed over time – e.g. our statistical analyses of tutorial interventions

included student identity as a factor or pretest score as a covariate item.

The third contribution this paper makes is on distinguishing between two effects of help: scaffolding immediate performance (scaffolding) vs. boosting persistent learning (learning) of help. Prior work assumed help has no direct impact on student learning (Conati, Gertner et al. 2002). Moreover, because we model tutor help and student model in one coherent framework, we can estimate the scaffolding and learning effect. This separation of immediate vs. persistent effect of help allows researchers to understand what the tutor intervention is really doing. For instance, it is possible to investigate whether some tutor interventions help persistent learning while others mainly help immediate performance.

Future Work and Conclusions

Currently, due to limitations in BNT-SM, we could only test models with discrete, binary variables. For example, in the *help* model, we only answer the question of *does help helps at all*. A more interesting question to ask is *which type of help helps better and when*. Thus, a future study is to extend BNT-SM to handle multi-nominal or even continuous variables, which allows modeling of different help types.

Another question that bothers us is why doesn't the *help* model fit the student performance data better then the KT model which ignores the tutor intervention data? We suggest a hypothesis that the *help* model has more parameter to estimate (8 parameters per skill and there are roughly 3000 skills) than the KT model (4 parameters per skill) and that the EM procedure may over fit the training set. Future study is needed to avoid this over fitting issue. One solution to this problem is to construct a hierarchical model where help has same impact on words.

Although our study has been mainly focused on modeling tutor help, our methodology can be applied to any kind of tutor intervention in general. Our work focuses on item level help (largely student initiated) because that is where we have the most data. This work is a challenging data mining problem because it requires a huge amount of data to estimate the model parameters reliably.

This paper describes an effort to measure the effectiveness of tutor help in an intelligent tutoring system. We propose a dynamic Bayes net to model tutor helps and student knowledge in one coherent framework. Even though the additional information in the *help* model does not yield a superior model fit, its informativeness outweighs the slight decrease in model fit. That is, the *help* model allows us to test evaluate the effectiveness of tutor intervention, which is essential to estimate the effectiveness of an ITS.

Acknowledgement

This work was supported by the National Science Foundation, ITR/IERI Grant No. REC-0326153. Any opinions, findings, and 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 also acknowledge members of Project LISTEN who contributed to the design and development of the Reading Tutor, and the schools that used the tutor.

References

Arroyo, I., J. E. Beck, C. R. Beal, R. E. Wing and B. P. Woolf (2001). Analyzing students' response to help provision in an elementary mathematics Intelligent Tutoring System. Help Provision and Help Seeking in Interactive Learning Environments. Workshop at the Tenth International Conference on Artificial Intelligence in Education., San Antonio, TX.

Beck, J. E., P. Jia and J. Mostow (2004). "Automatically assessing oral reading fluency in a computer tutor that listens." Technology, Instruction, Cognition and Learning **2**: 61-81.

Beck, J. E. and J. Sison (2004). Using knowledge tracing to measure student reading proficiencies. Proceedings of the 7th International Conference on Intelligent Tutoring Systems, Maceio, Brazil.

Chang, K., Beck, J. E., Mostow, J. and Corbett, A. (2006). A Bayes Net Toolkit for Student Modeling in Intelligent Tutoring Systems. Intelligent Tutoring Systems.

Cohen, P. R. (1995). Empirical Methods for Artificial Intelligence. Cambridge, Massachusetts, MIT Press.

Conati, C., A. Gertner and K. VanLehn (2002). "Using Bayesian Networks to Manage Uncertainty in Student Modeling." User Modeling and User-Adapted Interaction **12**(4): 371-417.

Corbett, A. and J. Anderson (1995). "Knowledge tracing: Modeling the acquisition of procedural knowledge." User modeling and user-adapted interaction **4**: 253-278.

Dean, T., Kanazawa, K. (1989). "A model for reasoning about persistence and causation." International Journal of Computational Intelligence **5**: 142-150.

Hand, D., H. Mannila and P. Smyth (2001). Principles of Data Mining. Cambridge, Massachusetts, MIT Press.

Heiner, C., J. E. Beck and J. Mostow (2004). Improving the Help Selection Policy in a Reading Tutor that Listens. Proceedings of the InSTIL/ICALL Symposium on NLP and Speech Technologies in

Advanced Language Learning Systems, Venice, Italy.
Mostow, J. and G. Aist (2001). Evaluating tutors that listen: An overview of Project LISTEN. Smart Machines in Education. P. Feltovich. Menlo Park, CA, MIT/AAAI Press**:** 169-234.
Murphy, K. (1998). Bayes Net Toolbox for Matlab. **2006**.
Reye, J. (2004). "Student modeling based on belief networks." International Journal of Artificial Intelligence in Education **14**: 1-33.