Learning a Tutorial Dialogue Policy for Delayed Feedback

Kristy Elizabeth Boyer, Robert Phillips*, Eun Young Ha, Michael D. Wallis*, Mladen A. Vouk, and James C. Lester

Department of Computer Science, North Carolina State University
*Dual affiliation with Applied Research Associates, Inc.
Raleigh, North Carolina, USA
Corresponding author: keboyer@ncsu.edu

Abstract

Creating natural language tutorial dialogue systems that realize effective strategies is a central challenge for intelligent tutoring systems research. Traditional approaches generally require large development time, do not generalize well across domains, and do not match the flexibility and natural language sophistication of human tutors. A promising approach that may offer several benefits is data-driven system development, in which a dialogue policy is learned from corpora of human tutorial dialogue. To date these learning approaches typically focus on optimizing the tutor's choice of act, and do not explicitly model the instances in which the tutor chose not to act. This paper reports on a hidden Markov modeling (HMM) approach within human textual tutorial dialogue that explicitly represents the tutors' choices not to intervene. The results show that an HMM that models tutor non-interventions predicts tutor moves significantly better than a model that does not explicitly represent the non-interventions. The findings have implications for automatically modeling tutorial strategies and for learning dialogue policies from corpora.

Introduction

Tutorial dialogue systems hold great promise for bringing individualized instruction with rich natural language dialogue to a broad population of students. Great strides have been made toward this goal, and a number of tutorial dialogue systems have been developed to date (e.g., Forbes-Riley & Litman, 2009; D'Mello et al., 2008; Dzikovska et al., 2010; VanLehn et al., 2007). Many of these systems are at the center of ongoing research projects that examine such factors as affective adaptation (Forbes-Riley & Litman, 2009; D'Mello et al., 2008), micro-level tutorial tactics (Chi, VanLehn, & Litman, 2010), and collaborative patterns (Rosé et al., 2008). Through such investigations and by leveraging rich empirical results, tutorial dialogue systems have grown in effectiveness and flexibility over time, and have contributed along the way to our understanding of fundamental processes of teaching and learning through tutoring.

While today's tutorial dialogue systems often achieve levels of effectiveness exceeding that of traditional classroom instruction (VanLehn et al., 2007), significant challenges remain. For example, creating tutoring system content traditionally requires substantial hand authoring efforts. Due in part to this challenge, tutorial dialogue systems still fall short of the effectiveness that has been observed with expert human tutors in a mastery learning situation (Bloom, 1984).

Machine learning approaches have the potential to address these challenges through more flexible, robust dialogue management. With a machine learning approach, a tutorial dialogue policy that prescribes the system's behavior is learned directly from a corpus of tutorial dialogue, rather than being manually authored. Such approaches have demonstrated great promise in ITS research for moving beyond traditional analysis and leveraging the increasing number of corpora that are available to tutoring systems researchers (Pardos, Dailey, & Heffernan, 2010; Barnes & Stamper, 2010; Chi, VanLehn, & Litman, 2010; Forbes-Riley & Litman, 2009; Dzikovska et al., 2010).

However, prior machine learning approaches focused on modeling the type of tutor action that should be taken, and not on the choice of *whether* to take action. In task-oriented tutoring, the choice to permit students to continue working without interruption is a viable one. The tradeoffs involved with immediate versus delayed feedback have been considered in a rich body of research (Shute, 2008), but to date no research has investigated how to create a computational model of tutor non-intervention as part of a fully machine-learned tutorial dialogue policy. This paper presents work toward that goal. We define two types of

tutor non-intervention: NO-REMEDIATION of buggy task actions (i.e., the tutor does not correct a student's error), and NO-MENTION of correctly completed subtasks (i.e., the tutor does not note the student's progress within a dialogue move). We report on a novel approach to discretizing and automatically tagging these events of tutor non-intervention within a corpus of tutorial dialogue. Additionally, we learn an HMM-based dialogue policy that accounts for the non-interventions, as evidenced by the learned models' ability to predict them within the corpus.

Related Work

Recent years have seen an increasing interest in data-driven approaches to intelligent tutoring systems research. Some of this work involves learning about the effectiveness of different tutoring strategies. For example, a recent line of investigation has proposed a method for testing the effectiveness of tutoring strategies from data, even when the data were not produced by randomized controlled trials (Pardos, Dailey, & Heffernan, 2010). The goal of this work is to automatically discover the most effective contextualized approaches from among those that an ITS already knows, and to adapt the ITS' behavior based on this new knowledge. Other work has leveraged reinforcement learning to compare the effectiveness of micro-level tactics, such as eliciting versus telling information, in given contexts (Chi, VanLehn, & Litman, 2010).

An important complementary line of investigation involves creating tutoring system behavior by learning it from a corpus. This is the goal of work that examines corpora of student problem-solving in a logic domain and uses prior students' success or failure to generate tutorial feedback during new student sessions (Barnes & Stamper, 2010). It has also been the focus of prior work to learn a dialogue policy for tutoring introductory computer science directly from an annotated corpus of human tutoring (Boyer et al., 2010a). In that work, it was found that hidden Markov models (HMMs) perform better than observable Markov, or bigram models, for predicting tutor dialogue acts within a corpus. The prediction accuracy was further improved by including a high-level notion of task progress.

The current work makes a step beyond the previous work by explicitly modeling tutor non-interventions as events within the observation sequences. Because the overall goal of the project is to capture human tutorial expertise within computational models of dialogue, representing the human tutor's choice *not* to take action is an important step. Such a model may illuminate strategic aspects of human tutors' approaches, and has the potential to be of great value for incorporating a learned dialogue policy in a runtime tutoring system.

Corpus and Annotation

The tutorial dialogue corpus was collected over the course of a human-human tutoring study in the domain of introductory computer programming. The 48 student participants were enrolled in a university introductory computer science course and were learning computer science concepts within the Java programming language. The two tutors had experience in tutoring computer science, but no formal training. Each student engaged in one task-oriented textual dialogue, lasting approximately one hour. On average, students exhibited statistically significant learning gains from pretest to posttest (paired *t*-test: $p<0.0001$). The students and tutors were located in separate rooms and worked together over the network with synchronized task views (Boyer et al., 2009). All task actions and textual dialogue were captured within a database. An excerpt of the corpus is displayed in Table 1.

Table 1. Corpus excerpt

Time	User	Dialogue Move or Task Action
10:50:31	Student	But I haven't been to strong with parameters *[NegContentFdbk]*
10:50:46	Stu. Task	p
10:50:46	Stu. Task	l
10:50:46	Stu. Task	o
10:50:46	Stu. Task	t
10:50:47	Stu. Task	T
10:50:47	Stu. Task	i
10:50:47	Stu. Task	m
10:50:47	Stu. Task	e
10:50:48	Stu. Task	s
10:50:48	Stu. Task	()
10:50:48	Stu. Task	;
10:52:37	Student	I'm not really sure why I'm gettin an error for plotValue but I'm pretty sure my parameter is wrong *[LkwmContentFdbk]*
10:52:54	Tutor	Yeah, it's because the parameter is wrong *[Stmt]*
10:53:05	Tutor	It takes a double.. and you're giving it an array *[Stmt]*

Dialogue Act Annotation

The dialogue act annotation scheme in Table 2 was applied to the entire corpus by a single annotator, and then independently applied to 10% of the corpus by a second annotator. The resulting Kappa statistic of 0.80 indicates very good reliability.

Table 2. Dialogue act tags

Dialogue Act	Student	Tutor	% St.	% Tu.[1]
ASSESSING QUESTION	*How does that declaration look?*	*Which type should that be?*	*.20*	*.09*
EXTRA-DOMAIN	*May I use my textbook?*	*A coordinator will be there soon.*	*.08*	*.03*
GROUNDING	*Got it.*	*Ok.*	*.26*	*.05*
LUKEWARM FDBK	*Sort of.*	*That's close.*	*.02*	*.02*
LUKEWARM CONTENT FDBK	*We covered arrays in class, but I'm not sure I understand them.*	*Almost there, but the second parameter isn't quite right.*	*.01*	*.03*
NEGATIVE FDBK	*No.*	*That's not right.*	*.04*	*.01*
NEGATIVE CONTENT FDBK	*I don't know how to initialize it.*	*No, the counter has to be an int.*	*.01*	*.08*
POSITIVE FDBK	*I understand.*	*Perfect.*	*.09*	*.13*
POSITIVE CONTENT FDBK	*Yes, the output matches.*	*Right, the array is a local variable.*	*.02*	*.02*
QUESTION	*How do I insert an element?*	*Which approach do you prefer?*	*.10*	*.02*
RESPONSE	*Go to 5.*	*It will be an int.*	*.09*	*.04*
STATEMENT	*We need an int.*	*They start at 0.*	*.07*	*.30*

Task Annotation

All task actions, which involved typing or deleting programming statements with the goal of solving the learning task, were generated by students and were logged at the keystroke level. Tutors viewed a synchronized problem-solving window but could not take problem-solving actions themselves. Each student task action was annotated along two axes: subtask structure and correctness. The manual subtask annotation first clustered keystrokes into semantic events at approximately the keyword level (e.g., *for* loop), then identified a high-level subtask (e.g., "Understand the Problem," "Module 1," "Module 2") and lower-level subtasks (e.g., "Write a for loop," "Declare an array"), with the final tag corresponding to a unique path along the task decomposition tree and an instance of Java syntax. A weighted Kappa statistic (Cohen, 1968) was used because weighted Kappa accounts for the distance between subtasks (e.g., subtask *1-a* is "closer" to subtask *1-b* than to *3-c*). The weighted Kappa statistic was 0.86 (Kappa 0.58) on 20% of the corpus.

Tagging the dialogue moves for subtask topic was a more challenging undertaking because unlike task actions to which the annotators were required to apply leaf tags from the task decomposition tree, for dialogue moves the annotators were permitted to apply a tag from any level of the tree. The combined weighted Kappa statistic for dialogue moves and task actions was 0.77 (Kappa 0.33). In addition to task structure annotation, each task action was labeled for correctness. The four labels are CORRECT, BUGGY, INCOMPLETE, or DISPREFERRED (technically functional but not adhering to the pedagogical goals of the learning task). The simple Kappa was 0.80.

Tutor Non-Intervention Events

After the dialogue acts and task actions were fully annotated, tutor non-interventions were identified within the corpus. In terms of time, events and non-events are heterogeneous entities: events are clearly associated with a discrete timestamp that can be logged, while non-events fill the continuous space around the events. Inspection of the corpus suggested an algorithm for identifying discrete events of tutor non-intervention (Figure 1).

```
Initialize list unremediatedBugs
Initialize list unmentionedComplete

For (all sessions s in corpus)

 For (all observations o in s)

   If o is a buggy task action

      1. If unremediatedBugs is
         non-empty, generate a NO-
         REMEDIATION event before o
      2. Clear unremediatedBugs
      3. Add o to unremediatedBugs

   Else if o is a correct task action

      1. Add o to list
         unmentionedComplete

   Else if o is a tutor feedback OR
           o is a tutor question
      1. Remove subtask to which o
         refers from list
         unremediatedBugs
      2. Remove subtask to which o
         refers from list
         unmentionedComplete
      4. If unremediatedBugs is
         non-empty, generate a NO-
         REMEDIATION event before o and
         clear unremediatedBugs
      3. Else if
         unmentionedComplete is
         non-empty, generate a NO-
         MENTION event before o and
         clear unmentionedComplete
```

Figure 1. Tutor non-intervention event algorithm

[1] Proportion of all tutor moves including the non-intervention moves of NO-REMEDIATION (.09) and NO-MENTION (.09).

In this algorithm, a NO-REMEDIATION event is generated when a bug that was introduced earlier by the student has not been remediated by the tutor before the student goes on to introduce another bug on a different subtask. NO-REMEDIATION events are also introduced if the tutor provides feedback or asks a question on some subtask, say subtask *a*, while a bug on another subtask, say subtask *a'*, was introduced earlier and has not yet been remediated. On the other hand, tutor NO-MENTION events capture the instances where a tutor does not mention a correctly completed subtask before moving on to provide feedback or ask a question regarding a different subtask.

This algorithm is designed to be conservative, inserting tutor non-intervention events at a restricted subset of all possible junctures where a non-intervention event may actually have occurred. For example, if there is more than one bug in the list of un-remediated bugs, only one NO-REMEDIATION event is generated (lines 11 and 30). This is to avoid inserting more than one NO-REMEDIATION event at a particular point in time, since the sequential modeling approach to be applied (HMMs) would be forced to treat these observations as occurring at different time steps. The same multi-event problem would be encountered if a tutor feedback move was permitted to trigger the generation of both a NO-REMEDIATION event and a NO-MENTION event, an occurrence that is prevented by giving priority to the former (hence the "else" of line 33).

Modeling Dialogue Structure with HMMs

The completed sequences of dialogue acts, task correctness labels, and tutor non-intervention events are given as input, with the goal of learning the dialogue policy. The machine learning framework utilizes HMMs to model sequences of annotated observations within the corpus. HMMs are a principled choice for this modeling task because they correspond well to widely accepted notions of the stochastic structure of dialogue, and have been shown to correlate with learning (Boyer et al., 2010a). The input sequences were preprocessed to join dependent adjacency pairs such as questions and responses, following an automatic adjacency-pair joining algorithm (Boyer et al., 2009).

HMMs explicitly model hidden states within a doubly stochastic structure (Rabiner, 1989). A first-order HMM, in which each hidden state depends only on the immediately preceding hidden state, is defined by the following components:

- $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_M\}$, the observation symbol alphabet
- $S = \{s_1, s_2, \ldots, s_N\}$, the set of hidden states
- $\Pi = [\pi_i]$, the initial probability distribution, where π_i is the probability of a sequence beginning in hidden state s_i in S
- $A = [a_{ij}]$, a transition probability distribution, where a_{ij} is the probability of the model transitioning from hidden state i to hidden state j
- $B = [b_{ik}]$, an emission probability distribution where b_{ik} is the probability of state i emitting observation symbol k.

In this work, the observation symbol alphabet Σ is given; it is defined by the union of all dialogue act tags, task correctness tags, and tutor non-intervention tags. The transition probability distribution A, emission probability distribution B, and initial probability distribution Π are learned by the standard Baum-Welch algorithm for optimizing HMM parameters (Rabiner, 1989), with five-time random restart that included new initial parameters for each model to reduce the probability of selecting a model that represents only a local optimum. The number of hidden states N is selected by running the full HMM training algorithm including random restarts in ten-fold cross-validation across the data and selecting the model size with the best mean Bayesian Information Criterion, which penalizes the log likelihood pursuant to the number of parameters.

Learned HMM

The process described above was applied to learn a best-fit model, which has N=4 hidden states. The emission probability distribution of these hidden states is depicted in Figure 2, with observation symbols whose probability was less than 0.05 omitted for simplicity.

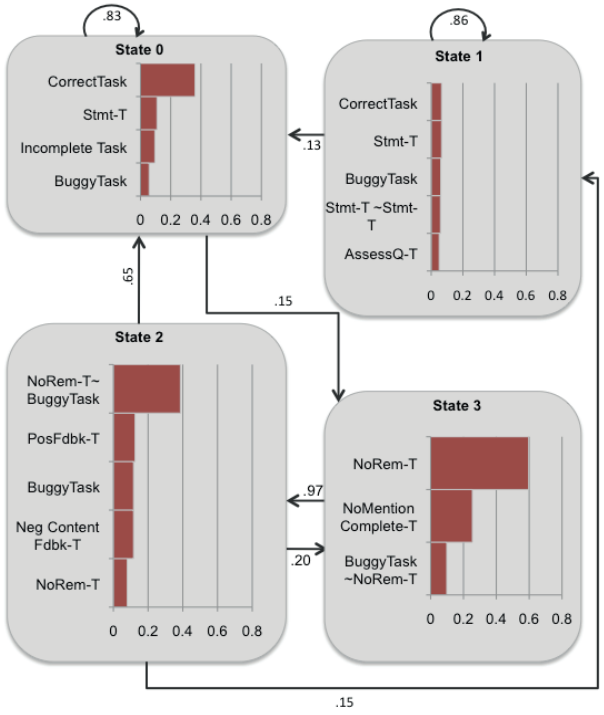

Figure 2. HMM that models tutor non-interventions. Emission and transition probabilities ≥ 0.05 shown

From State 0, which is dominated by student correct task actions, the model is most likely to make a self-transition back to this state; that is, students working correctly are likely to continue to do so. From this state, the model transitions to State 3 with probability 0.15. State 3 emits tutor non-intervention moves of NO-REMEDIATION and NO-MENTION, and also emits pairs of buggy task actions and tutor non-remediations. This transition suggests that particularly when the student has been working correctly before, tutors are less likely to intervene at the first occurrence of a bug in the task. State 1 reflects activities typically associated with teaching and evaluation by the tutor, with a high probability of self-transition, while State 2 includes active remediation and is likely to transition back to the correct student work state.

Prediction Accuracy for Tutor Moves

To determine how well the learned model captures the dialogue policy implicitly represented by the corpus, we measure the accuracy with which the model predicts human tutor dialogue moves and non-interventions. To assess the impact on the model of adding tutor non-intervention events, we conducted the prediction experiment with two versions of the model: one with tutor non-intervention events, and the other without. Figure 3 displays the prediction accuracy of the two models. Note that although the model that includes tutor non-interventions must discriminate among more categories, it performs slightly better on nearly all training folds than the model that does not include tutor non-intervention events.

The majority class baseline accuracy for the model that includes tutor non-interventions is 0.30, which is the relative frequency of tutor statements among all tutor moves (including non-interventions). The corresponding baseline accuracy for the model that does not include non-interventions is 0.38, the relative frequency of tutor statements among only dialogue acts.

The prediction accuracies in Figure 3 demonstrate that the additional categories of NO-REMEDIATION and NO-MENTION do not adversely affect the model's ability to predict tutor moves. In fact, the average prediction accuracy for the model that includes non-interventions was 0.55, significantly higher than the average prediction accuracy for the dialogue-move-only model of 0.53 (two-tailed paired *t*-test: p=0.027). This improvement is likely due in part to the fact that when considering dialogue moves only, periods of "silence" were not modeled, and in fact, those periods almost certainly influenced subsequent conversation.

While the HMM clearly performs above a majority class baseline, it is also meaningful to consider how well the model performs compared to a less trivial baseline. Specifically, we compared the HMM dialogue policy to an observable Markov, or bigram, dialogue policy under the same two conditions described above. A first-order observable Markov model is simpler than an HMM, defined fully by the observation symbol alphabet Σ and the transition probability distribution A among the symbols. The results of the comparison are shown in Figure 4. The average prediction accuracy of the bigram model was 0.34, significantly lower than the average of 0.55 for the HMM (two-tailed paired *t*-test: p<0.00001). These findings are consistent with those in previous work, in which HMMs performed similarly better than bigram models on this task when tutor non-interventions were not considered (Boyer et al., 2010b).

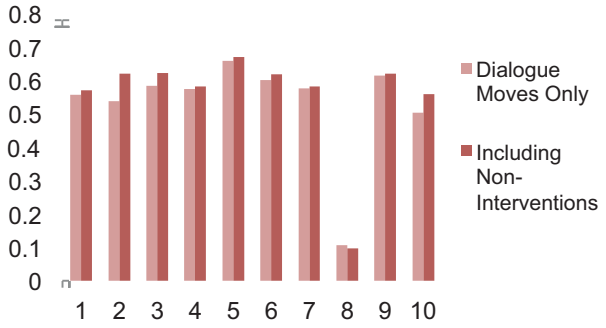

Figure 3. HMM prediction accuracy vs. cross-validation fold for tutor move prediction

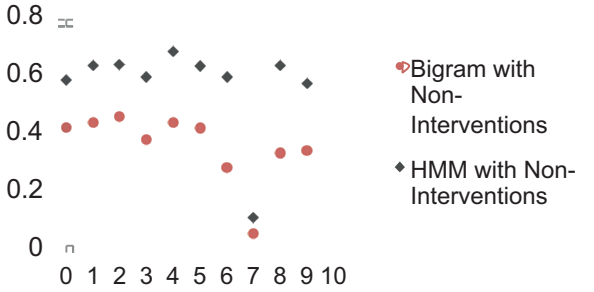

Figure 4. Prediction accuracy vs. cross-validation fold for HMM dialogue policy and bigram dialogue policy

Conclusion and Future Work

Human tutoring features rich natural language dialogue and generally incorporates highly flexible tutorial dialogue policies. Creating a tutoring system capable of such richness and flexibility poses many challenges. A promising approach involves learning dialogue policies directly from corpora of human tutoring. This paper has presented a machine learning approach that explicitly models tutors' choices not to intervene. The learned HMM predicts tutor dialogue moves and non-interventions within a corpus of task-oriented tutorial dialogue. The two tutor non-intervention moves were NO-REMEDIATION of student buggy task actions, and NO-MENTION of correctly completed student task actions. For comparison, an HMM was also learned without these tutor non-intervention events, and its prediction accuracy was lower than the

prediction accuracy of the model that included the non-interventions (55% accuracy vs. 53% accuracy; p=0.027). The experiments using this simplified model of delayed feedback indicate that tutor non-interventions are desirable to model within task-oriented dialogue, where choosing not to interrupt the student's problem solving is often a viable strategy. The prediction accuracy results suggest that these non-intervention events can be handled well by a learned dialogue policy model, and that the periods of "silence" may impact subsequent dialogue.

There are several notable limitations to the current work. First, the tutoring corpus reflects non-expert tutoring, which is known to have many important differences from expert tutoring. Applying the techniques to corpora of expert tutoring is an important direction for future work. Secondly, the algorithm for identifying tutor non-interventions is conservative, and as such probably omits some non-intervention tags that occurred in reality. Finally, the regularities introduced by the deterministic algorithm may facilitate model training and inflate the prediction accuracy, a possibility that should be further explored.

This investigation highlights some important directions for future work. First, learned dialogue models must take advantage of more detailed task information and timing data; the current work has only scratched the surface of these sources of information. Additionally, the time required to manually annotate the dialogue and task actions is high, and research toward unsupervised dialogue act tagging and unsupervised plan recognition is of paramount importance as the field moves toward data-driven creation of tutorial dialogue systems.

Acknowledgments. The authors wish to thank Sidney D'Mello, Danielle McNamara, Andrew Olney, and Natalie Person for conversations that inspired this analysis. This work is supported in part by the NC State University Department of Computer Science along with the National Science Foundation through Grants REC-0632450, IIS-0812291, DRL-1007962 and the STARS Alliance Grant CNS-0739216. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

References

Barnes, T., & Stamper, J. (2010). Automatic hint generation for logic proof tutoring using historical data. *Proceedings of ITS 2010,* Pittsburgh, PA. 3-14.

Bloom, B. S. (1984). The 2 Sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13*(6), 4-16.

Boyer, K.E., Phillips, R., Ingram, A., Ha, E.Y., Wallis, M.D., Vouk, M.A., & Lester, J. (2010a). Characterizing the effectiveness of tutorial dialogue with hidden Markov models. *Proceedings of ITS 2010,* 55-64.

Boyer, K.E., Phillips, R., Ha, E.Y., Wallis, M.D., Vouk, M.A., & Lester, J. (2010b). Leveraging hidden dialogue state to select tutorial moves. *Proceedings of the NAACL HLT Workshop on Innovative Use of NLP for Building Educational Applications,* 66-73.

Boyer, K.E., Phillips, R., Ha, E.Y., Wallis, M.D., Vouk, M.A., & Lester, J. (2009). Modeling dialogue structure with adjacency pair analysis and hidden Markov models. *Proceedings of NAACL HLT Short Papers,* 49-52.

Chi, M., VanLehn, K., & Litman, D. (2010). Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. *Proceedings ITS 2010,* 224-234.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin, 70*(4), 213-220.

D'Mello, S., Jackson, T., Craig, S., Morgan, B., Chipman, P., White, H., Person, N., Kort, B., el Kaliouby, R., & Picard, R. (2008). AutoTutor detects and responds to learners' affective and cognitive states. *Proceedings of the Workshop on Emotional and Cognitive Issues in ITS in Conjunction with ITS 2008,* 31-43.

Dzikovska, M. O., Moore, J. D., Steinhauser, N., Campbell, G., Farrow, E., & Callaway, C. B. (2010). BEETLE II: A system for tutoring and computational linguistics experimentation. *Proceedings of Association for Computational Linguistics (ACL) System Demonstrations Track,* 13-18.

Forbes-Riley, K., & Litman, D. (2009). Adapting to student uncertainty improves tutoring dialogues. *Proceedings of AIED 2009,* 33-40.

Pardos, Z., Dailey, M., & Heffernan, N. (2010). Learning what works in ITS from non-traditional randomized controlled trial data. *Proceedings of ITS 2010,* 41-50.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257-286.

Rosé, C., Wang, Y. C., Cui, Y., Arguello, J., Stegmann, K., Weinberger, A., & Fischer, F. (2008). Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in computer-supported collaborative learning. *International Journal of Computer-Supported Collaborative Learning, 3*(3), 237-271.

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research, 78*(1), 153.

VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rosé, C. P. (2007). When are tutorial dialogues more effective than reading? *Cognitive Science, 31*(1), 3-62.