Using Reinforcement Learning to Build a Better Model of Dialogue State

Joel R. Tetreault
University of Pittsburgh
Learning Research and Development Center
Pittsburgh PA, 15260, USA
tetreaul@pitt.edu

Diane J. Litman
University of Pittsburgh
Department of Computer Science &
Learning Research and Development Center
Pittsburgh PA, 15260, USA
litman@cs.pitt.edu

Abstract

Given the growing complexity of tasks that spoken dialogue systems are trying to handle, Reinforcement Learning (RL) has been increasingly used as a way of automatically learning the best policy for a system to make. While most work has focused on generating better policies for a dialogue manager, very little work has been done in using RL to construct a better dialogue state. This paper presents a RL approach for determining what dialogue features are important to a spoken dialogue tutoring system. Our experiments show that incorporating dialogue factors such as dialogue acts, emotion, repeated concepts and performance play a significant role in tutoring and should be taken into account when designing dialogue systems.

1 Introduction

This paper presents initial research toward the long-term goal of designing a tutoring system that can effectively adapt to the student. While most work in Markov Decision Processes (MDPs) and spoken dialogue have focused on building better policies (Walker, 2000; Henderson et al., 2005), to date very little empirical work has tested the utility of adding specialized features to construct a better dialogue state. We wish to show that adding more complex factors to a representation of student state is a worthwhile pursuit, since it alters what action the tutor should make. The five dialogue factors we explore are *dialogue acts*, *certainty level*, *frustration level*, *concept repetition*, and *student performance*. All five are factors that are not just unique to the tutoring domain but are important to dialogue systems in general. Our results show that using these features, combined with the common baseline of student correctness, leads to a significant change in the policies produced, and thus should be taken into account when designing a system.

2 Background

We follow past lines of research (such as (Singh et al., 1999)) for describing a dialogue d as a trajectory within a Markov Decision Process (Sutton and Barto, 1998). A MDP has four main components: *states*, *actions*, a *policy*, which specifies what is the best action to take in a state, and a *reward function* which specifies the utility of each state and the process as a whole. Dialogue management is easily described using a MDP because one can consider the actions as actions made by the system, the state as the dialogue context, and a reward which for many dialogue systems tends to be task completion success or dialogue length. Typically the state is viewed as a vector of features such as dialogue history, speech recognition confidence, etc.

The goal of using MDPs is to determine the best policy π for a certain state and action space. That is, we wish to find the best combination of states and actions to maximize the reward at the end of the dialogue. In most dialogues, the exact reward for each state is not known immediately, in fact, usually only the final reward is known at the end of the dialogue. As long as we have a reward function, Reinforcement Learning allows one to automatically compute the best policy. The following recursive equation gives us a way of calculating the expected cumulative value (V-value) of a state s (-value):

$$V(s) = \sum_{s'} P_{ss'}^{\pi(s)}[R_{ss'}^{\pi(s)} + \gamma V(s')]$$

Here $\pi(s)$ is the best action for state s at this time, P is the probability of getting from state s to s' via $\pi(s)$. This is multiplied by the sum of the reward R for that traversal plus the value of the new state multiplied by a discount factor γ. γ ranges between 0 and 1 and discounts the value of past states. The *policy iteration* algorithm (Sutton and Barto, 1998) iteratively updates the value of each state V(s) based on the values of its neighboring states. The iteration stops when each update yields an epsilon difference (implying that V(s) has converged) and we select the action that produces the highest V-value for that state.

Normally one would want a dialogue system to interact with users thousands of times to explore the entire traversal space of the MDP, however in practice that is very time-consuming. Instead, the next best tactic is to train the MDP (that is, calculate transition probabilities for getting from one state to another, and the reward for each state) on already collected data. Of course, the whole space will not be considered, but if one reduces the size of the state vector effectively, data size becomes less of an issue (Singh et al., 2002).

3 Corpus

For our study, we used an annotated corpus of 20 human-computer spoken dialogue tutoring sessions. Each session consists of an interaction with one student over 5 different college-level physics problems, for a total of 100 dialogues. Before the 5 problems, the student is asked to read physics material for 30 minutes and then take a pre-test based on that material. Each problem begins with the student writing out a short essay response to the question posed by the computer tutor. The system reads the essay and detects the problem areas and then starts a dialogue with the student asking questions regarding the confused concepts. Informally, the dialogue follows a question-answer format. Each of the dialogues has been manually authored in advance meaning that the system has a response based on the correctness of the student's last answer. Once the student has successfully answered all the questions, he or she is asked to correct the initial essay. On average, each of the dialogues takes 20 minutes and contains 25 student turns. Finally, the student is given a post-test similar to the pre-test, from which we can calculate their normalized learning gain:

$$NLG = \frac{post - pre}{1 - pre}$$

Prior to our study, the corpus was then annotated for Student and Tutor Moves (see Tables 1 and 2) which can be viewed as Dialogue Acts (Forbes-Riley et al., 2005). Note that tutor and student turns can consist of multiple utterances and can thus be labeled with multiple moves. For example, a tutor can give feedback and then ask a question in the same turn. Whether to include feedback will be the action choice addressed in this paper since it is an interesting open question in the Intelligent Tutoring Systems (ITS) community. Student Moves refer to the type of answer a student gives. Answers that involve a concept already introduced in the dialogue are called Shallow, answers that involve a novel concept are called Novel, "I don't know" type answers are called Assertions (As), and Deep answers refer to answers that involve linking two concepts through reasoning. In our study, we merge all non-Shallow moves into a new move "Other."

In addition to Student Moves, we annotated five other features to include in our representation of the student state. Two emotion related features were annotated manually (Forbes-Riley and Litman, 2005): certainty and frustration. Certainty describes how confident a student seemed to be in his answer, while frustration describes how frustrated the student seemed to be in his last response. We include three other features for the Student state that were extracted automatically. Correctness says if the last student answer was correct or incorrect. As noted above, this is what most current tutoring systems use as their state. Percent Correct is the percentage of questions in the current problem the student has answered correctly so far. Finally, if a student performs poorly when it comes to a certain topic, the system may be forced to repeat a description of that concept again (concept repetition).

It should be noted that all the dialogues were authored beforehand by physics experts. For every turn there is a list of possible correct, incorrect and partially correct answers the student can make, and then for each one of these student responses a link to the next turn. In addition to

State	Parameters
Student Move	Shallow (S) Novel & As & Deep (O)
Certainty	Certain, Uncertain, Neutral
Frustration	Frustrated (F), Neutral (N),
Correctness	Correct (C), Incorrect (I) Partially Correct (PC)
Percent Correct	50-100% (High), 0-50% (Low)
Concept Repetition	Concept is not repeated (0), Concept is repeated (R)

Table 1: Student Features in Tutoring Corpus

Action	Parameters
Tutor Feedback Act	Positive, Negative
Tutor Question Act	Short Answer Question (SAQ) Complex Answer Question (CAQ)
Tutor State Act	Restatement, Recap, Hint Expansion, Bottom Out

Table 2: Tutor Acts in Tutoring Corpus

explaining physics concepts, the authors also include feedback and other types of helpful measures (such as hints or restatements) to help the student along. These were not written with the goal of how best to influence student state. Our goal in this study is to automatically learn from this corpus which state-action patterns evoke the highest learning gain.

4 Infrastructure

To test different hypotheses of what features best approximate the student state and what are the best actions for a tutor to consider, one must have a flexible system that allows one to easily test different configurations of states and actions. To accomplish this, we designed a system similar to the Reinforcement Learning for Dialogue Systems (RLDS) (Singh et al., 1999). The system allows a system designer to specify what features will compose the state and actions as well as perform operations on each individual feature. For instance, the tool allows the user to collapse features together (such as collapsing all Question Acts together into one) or quantize features that have continuous values (such as the number of utterances in the dialogue so far). These collapsing functions allow the user to easily constrain the trajectory space. To further reduce the search space for the MDP, our tool allows the user to specify a threshold to combine states that occur less than the threshold into a single "threshold state." In addition, the user can specify a reward function and a discount factor,

For this study, we use a threshold of 50 and a discount factor of 0.9, which is also what is commonly used in other RL models, such as (Frampton and Lemon, 2005). For the dialogue reward function, we did a median split on the 20 students based on their normalized learning gain, which is a standard evaluation metric in the Intelligent Tutoring Systems community. So 10 students and their respective 5 dialogues were assigned a positive reward of +100 (high learners), and the other 10 students and their respective dialogues were assigned a negative reward of -100 (low learners). It should be noted that a student's 5 dialogues were assigned the same reward since there was no way to approximate their learning gain in the middle of a session.

The output of the tool is a probability matrix over the user-specified states and actions. This matrix is then passed to an MDP toolkit (Chades et al., 2005) written in Matlab.[1] The toolkit performs policy iteration and generates a policy as well as a list of V-values for each state.

5 Experimental Method

With the infrastructure created and the MDP parameters set, we can then move on to the goal of this experiment - to see what sources of information impact a tutoring dialogue system. First, we need to develop a baseline to compare the effects of adding more information. Second, we generate a new policy by adding the new information source to the baseline state. However, since we are currently not running any new experiments to test our policy, or evaluating over user simulations, we evaluate the reliability of our policies by looking at how well they converge over time, that is, if you incrementally add more data (ie. a student's 5 dialogues) does the policy generated tend to stabilize over time? And also, do the V-values for each state stabilize over time as well? The intuition is that if both the policies and V-values tend to converge then we can be sure that the policy generated is reasonable.

The first step in our experiment is to determine a baseline. We use feedback as our system action in our MDP. The action size is 3 (tutor can give feedback (Feed), give feedback with another tutor act (Mix), or give no feedback at all (NonFeed). Examples from our corpus can be seen in Table 3. It should be noted that "NonFeed" does not mean that the student's answer is not acknowledged, it

[1]MDP toolkit can be downloaded from http://www.inra.fr/bia/T/MDPtoolbox/

Case	Tutor Moves	Example Turn
Feed	Pos	'Super."
Mix	Pos, SAQ	'Good. What is the direction of that force relative to your fi st?"
NonFeed	Hint, CAQ	"To analyze the pumpkin's acceleration we will use Newton's Second Law. What is the defi nition of the law?"

Table 3: Tutor Action Examples

means that something more complex than a simple positive or negative phrase is given (such as a Hint or Restatement). Currently, the system's response to a student depends only on whether or not the student answered the last question correctly, so we use correctness as the sole feature in our dialogue state. Recall that a student can either be correct, partially correct, or incorrect. Since partially correct occurs infrequently compared to the other two, we reduced the state size to two by combining Incorrect and Partially Correct into one state (IPC) and keeping correct (C).

The third column of Table 4 has the resulting learned MDP policy as well as the frequencies of both states in the data. So for both states, the best action for the tutor to make is to give feedback, without knowing anything else about the student state.

The second step in our experiment is to test whether the policies generated are indeed reliable. Normally, the best way to verify a policy is by conducting experiments and seeing if the new policy leads to a higher reward for the new dialogues. In our context, this would entail running more subjects with the augmented dialogue manager and checking if the students had a higher learning gain with the new policies. However, collecting data in this fashion can take months. So, we take a different tact of checking if the polices and values for each state are indeed converging as we add data to our MDP model. The intuition here is that if both of those parameters were varying between a corpus of 19 students to 20 students, then we can't assume that our policy is stable, and hence not reliable. However, if these parameters converged as more data was added, this would indicate that the MDP is reliable.

To test this out, we conducted a 20-fold cross-averaging test over our corpus of 20 students. Specifically, we made 20 random orderings of our students to prevent any one ordering from giving a false convergence. Each ordering was then chunked into 20 cuts ranging from a size of 1 student, to the entire corpus of 20 students. We then passed each cut to our MDP infrastructure such that we started with a corpus of just the first student of the ordering and then determined a MDP policy for that cut, then added another student to that original corpus and reran our MDP system. We continue this incremental addition of a student (5 dialogues) until we completed all 20 students. So at the end, we have 20 random orderings with 20 cuts each, so 400 MDP trials were run. Finally, we average the V-values of same size cuts together to produce an average V-value for that cut size. The left-hand graph in Figure 1 shows a plot of the average V-values for each state against a cut. The state with the plusses is the positive final state, and the one at the bottom is the negative final state. However, we are most concerned with how the non-final states converge, which are the states in the middle. The plot shows that for early cuts, there is a lot of instability but then each state tends to stabilize after cut 10. So this tells us that the V-values are fairly stable and thus reliable when we derive policies from the entire corpus of 20 students.

As a further test, we also check that the policies generated for each cut tend to stabilize over time. That is, the differences between a policy at a smaller cut and the final cut converge to zero as more data is added. This "diffs" test is discussed in more detail in Section 6.

6 Results

In this section, we investigate whether adding more information to our student state will lead to interesting policy changes. First, we add certainty to our baseline of correctness, and then compare this new baseline's policy (henceforth Baseline 2) with the policies generated when student moves, frustration, concept repetition, and percent correctness are included. For each test, we employed the same methodology as with the baseline case of doing a 20-fold cross-averaging and examining if the states' V-values converge.

We first add certainty to correctness because prior work (such as (Bhatt et al., 2004)) has shown the importance of considering certainty in tutoring

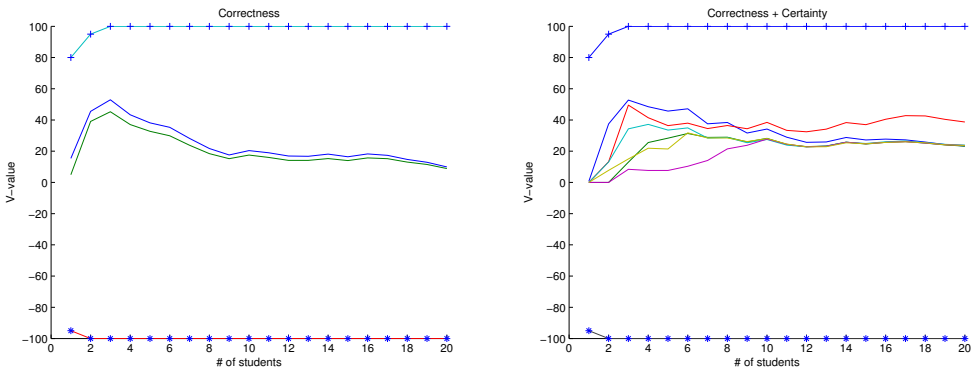

Figure 1: Baseline 1 and Baseline 2 Convergence Plots

systems. For example, a student who is correct and certain probably does not need a lot of feedback. But one that is correct but uncertain could signal that the student is becoming doubtful or at least confused about a concept. There are three types of certainty: certain (cer), uncertain (unc), and neutral (neu). Adding these to our state representation increases state size from 2 to 6. The new policy is shown in Table 4. The second and third columns show the original baseline states and their policies. The next column shows the new policy when splitting the original state into the new three states based on certainty, as well as the frequency of the new state. So the first row can be interpreted as if the student is correct and certain, one should give no feedback; if the student is correct and neutral, give feedback; and if the student is correct and uncertain, give non-feedback.

#	State	Baseline	+Certainty
1	C	**Feed** (1308)	cer: **NonFeed** (663) neu: Feed (480) unc: **NonFeed** (165)
2	IPC	**Feed** (872)	cer: **NonFeed** (251) neu: **Mix** (377) unc: **NonFeed** (244)

Table 4: Baseline Policies

Our reasoning is that if a feature is important to include in a state representation it should change the policies of the old states. For example, if certainty did not impact how well students learned (as deemed by the MDP) then the policies for certainty, uncertainty, and neutral would be the same as the original policy for Correct or Incorrect/Partially Correct, in this case they would be Feed. However, the figures show otherwise as when you add certainty to the state, only one new state (C while being neutral) retains the old policy of having the tutor give feedback. The policies which differ with the original are shown in bold.

So in general, the learned policy is that one should not give feedback if the student is certain or uncertain, but rather give some other form of feedback such as a Hint or a Restatement perhaps. But when the student is neutral with respect to certainty, one should give feedback. One way of interpreting these results is that given our domain, for students who are confident or not confident at all in their last answer, there are better things to say to improve their learning down the road than "Great Job!" But if the student does not display a lot of emotion, than one should use explicit positive or negative feedback to perhaps bolster their confidence level.

The right hand graph in Figure 1 shows the convergence plot for the baseline state with certainty. It shows that as we add more data, the values for each state converge. So in general, we can say that the values for our Baseline 2 case are fairly stable.

Next, we add Student Moves, Frustration, Concept Repetition, and Percent Correct features individually to Baseline 2. The first graph in Figure 2 shows a plot of the convergence values for the Percent Correct feature. We only show one convergence plot since the other three are similar. The result is that the V-values for all four converge after 14-15 students.

The second graph shows the differences in policies between the final cut of 20 students and all smaller cuts. This check is necessary because some states may exhibit stable V-values but actually be oscillating between two different policies of equal values. So each point on the graph tells us how many differences in policies there are between the cut in question and the final cut. For

example, if the policy generated at cut 15 was to give feedback for all states, and the policy at the final cut was to give feedback for all but two states, the "diff" for cut 15 would be two. So in the best case, zero differences mean that the policies generated for both cuts are exactly the same. The diff plots shows the differences decrease as data is added and they exhibited very similar plots to both Baseline cases. For cuts greater than 15, there are still some differences but these are usually due to low frequency states. So we can conclude that since our policies are fairly stable they are worth investigating in more detail.

In the remainder of this section, we look at the differences between the Baseline 2 policies and the policies generated by adding a new feature to the Baseline 2 state. If adding a new feature actually does not really change what the tutor should do (that is, the tutor will do the baseline policy regardless of the new information), one can conclude that the feature is not worth including in a student state. On the other hand, if adding the state results in a much different policy, then the feature is important to student modeling.

Student Move Feature The results of adding Student Moves to Baseline 2 are shown in Table 5. Out of the 12 new states created, 7 deviate from the original policy. The main trend is for the neutral and uncertain states to give mixed feedback after a student shallow move, and a non-feed response when the student says something deep or novel. When the student is certain, always give a mixed response except in the case where he said something Shallow and Correct.

#	State	Baseline	New Policy
1	certain:C	**NonFeed**	S: NonFeed O: **Mix**
2	certain:IPC	**NonFeed**	S: **Mix** O: **Mix**
3	neutral:C	**Feed**	S: Feed O: **NonFeed**
4	neutral:IPC	**Mix**	S: Mix O: **NonFeed**
5	uncertain:C	**NonFeed**	S: **Mix** O: NonFeed
6	uncertain:IPC	**NonFeed**	S: **Mix** O: NonFeed

Table 5: Student Move Policies

Concept Repetition Feature Table 6 shows the new policy generated. Unlike the Student Move policies which impacted all 6 of the baseline states, Concept Repetition changes the policies for the first three baseline states resulting in 4 out of 12 new states differing from the baseline. For states 1 through 4, the trend is that if the concept has been repeated, the tutor should give feedback or a combination of feedback with another Tutor Act. Intuitively, this seems clear because if a concept were repeated it shows the student is not understanding the concept completely and it is neccessary to give them a little more feedback than when they first see the concept. So, this test indicates that keeping track of repeated concepts has a significant impact on the policy generated.

#	State	Baseline	New Policy
1	certain:C	**NonFeed**	0: NonFeed R: **Feed**
2	certain:IPC	**NonFeed**	0: **Mix** R: **Mix**
3	neutral:C	**Feed**	0: **Mix** R: Feed
4	neutral:IPC	Mix	0: Mix R: Mix
5	uncertain:C	NonFeed	0: NonFeed R: NonFeed
6	uncertain:IPC	NonFeed	0: NonFeed R: NonFeed

Table 6: Concept Repetition Policies

Frustration Feature Table 7 shows the new policy generated. Comparing the baseline policy with the new policy (which includes categories for when the original state is either neutral or frustration), shows that adding frustration changes the policy for state 1, when the student is certain or correct. In that case, the better option is to give them positive feedback. For all other states, frustration occurs with each of them so infrequently [2] that the resulting states appeared less than the our threshold of 50 instances. As a result, these 5 frustration states are grouped together in the "threshold state" and our MDP found that the best policy when in that state is to give no feedback. So the two neutral states change when the student is frustrated. Interestingly, for students that are uncertain, the policy does not change if they are frustrated or neutral. The trend is to always give NonFeedback.

Percent Correctness Feature Table 8 shows the new policy generated for incorporating a simple model of current student performance within the dialog. This feature, along with Frustration, seems to impact the baseline the state least since both only alter the policies for 3 of the 12 new

[2] Only 225 out of 2180 student turns are marked as frustration, while all the others are neutral

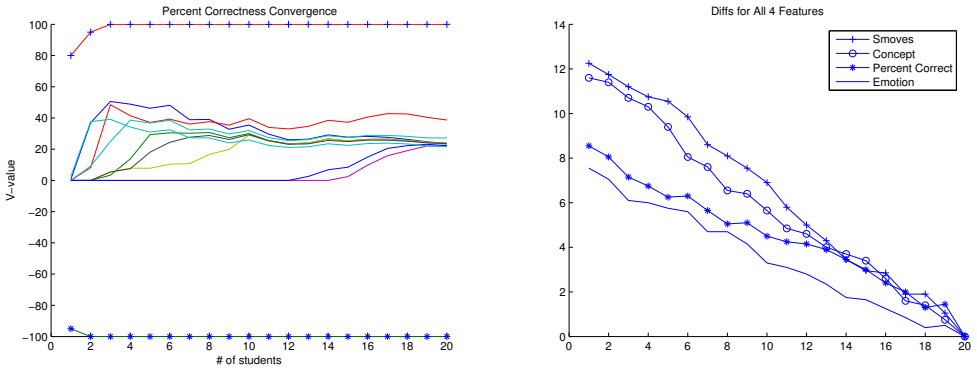

Figure 2: Percent Correct Convergence, and Diff Plots for all 4 Features

#	State	Baseline	New Policy
1	certain:C	**NonFeed**	N: NonFeed F: **Feed**
2	certain:IPC	NonFeed	N: NonFeed F: NonFeed
3	neutral:C	**Feed**	N: Feed F: **NonFeed**
4	neutral:IPC	**Mix**	N: Mix F: **NonFeed**
5	uncertain:C	NonFeed	N: NonFeed F: NonFeed
6	uncertain:IPC	NonFeed	N: NonFeed F: NonFeed

Table 7: Frustration Policies

states. States 3, 4, and 5 show a change in policy for different parameters of correctness. One trend seems to be that when a student has not been performing well (L), to give a NonFeedback response such as a hint or restatement.

#	State	Baseline	New Policy
1	certain:C	NonFeed	H: NonFeed L: NonFeed
2	certain:IPC	NonFeed	H: NonFeed L: NonFeed
3	neutral:C	**Feed**	H: Feed L: **NonFeed**
4	neutral:IPC	**Mix**	H: Mix L: **NonFeed**
5	uncertain:C	**NonFeed**	H: **Mix** L: NonFeed
6	uncertain:IPC	NonFeed	H: NonFeed L: NonFeed

Table 8: % Correctness Policies

7 Related Work

RL has been applied to improve dialogue systems in past work but very few approaches have looked at which features are important to include in the dialogue state. (Paek and Chickering, 2005) showed how the state space can be learned from data along with the policy. One result is that a state space can be constrained by only using features that are relevant to receiving a reward. Singh et al. (1999) found an optimal dialogue length in their domain, and showed that the number of information and distress attributes impact the state. They take a different approach than the work here in that they compare which feature values are optimal for different points in the dialogue. Frampton et al. (2005) is similar to ours in that they experiment on including another dialogue feature into their baseline system: the user's last dialogue act, which was found to produce a 52% increase in average reward. Williams et al. (2003) used Supervised Learning to select good state and action features as an initial policy to bootstrap a RL-based dialoge system. They found that their automatically created state and action seeds outperformed hand-crafted polices in a driving directions corpus. In addition, there has been extensive work on creating new corpora via user simulations (such as (Georgila et al., 2005)) to get around the possible issue of not having enough data to train on. Our results here indicate that a small training corpus is actually acceptable to use in a MDP framework as long as the state and action features are pruned effectively. The use of features such as context and student moves is nothing new to the ITS community however, such as the BEETLE system (Zinn et al., 2005), but very little work has been done using RL in developing tutoring systems.

8 Discussion

In this paper we showed that incorporating more information into a representation of the student state has an impact on what actions the tutor should take. We first showed that despite not be-

ing able to test on real users or simulated users just yet, that our generated policies were indeed reliable since they converged in terms of the V-values of each state and the policy for each state.

Next, we showed that all five features investigated in this study were indeed important to include when constructing an estimation of the student state. Student Moves, Certainty and Concept Repetition were the most compelling since adding them to their respective baseline states resulted in major policy changes. Tracking the student's frustration levels and how correct the student had been in the dialogue had the least impact on policies.

While these features (and their resulting policies) may appear unique to tutoring systems they also generalize to dialogue systems as a whole. Repeating a concept (whether it be a physics term or travel information) is important because it is an implicit signal that there might be some confusion and a different action is needed when the concept is repeated. Whether a student (or user) gives a short answer or a good explanation can indicate to the system how well the user is understanding system questions. Emotion detection and adaptation is a key issue for any spoken dialogue systems as designers try to make the system as easy to use for a student or trip-planner, etc. Frustration can come from difficulty in questions or in the more frequent problem for any dialogue system, speech recognition errors, so the manner in dealing with it will always be important. Percent Correctness can be viewed as a specific instance of tracking user performance such as if they are continously answering questions properly or are confused by what the system wants from them.

In terms of future work, we are currently annotating more human-computer dialogue data and will triple the size of our test corpus allowing us to 1. create more complicated states since more states will have been explored and 2. test out more complex tutor actions such as when to give Hints and Restatements. Finally, we are in the process of running this same experiment on a corpus of human-human tutoring dialogues to compare if human tutors have different policies.

9 Acknowledgments

We would like to thank the ITSPOKE group and the three anonymous reviewers for their insight and comments. Support for the research reported in this paper was provided by NSF grants #0325054 and #0328431.

References

K. Bhatt, M. Evens, and S. Argamon. 2004. Hedged responses and expressions of affect in human/human and human computer tutorial interactions. In *Proc. Cognitive Science*.

I. Chades, M. Cros, F. Garcia, and R. Sabbadin. 2005. Mdp toolbox v2.0 for matlab.

K. Forbes-Riley and D. Litman. 2005. Using bigrams to identify relationships between student certainness states and tutor responses in a spoken dialogue corpus. In *SIGDial*.

K. Forbes-Riley, D. Litman, A. Huettner, and A. Ward. 2005. Dialogue-learning correlations in spoken dialogue tutoring. In *AIED*.

M. Frampton and O. Lemon. 2005. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*.

K. Georgila, J. Henderson, and O. Lemon. 2005. Learning user simulations for information state update dialogue systems. In *Interspeech*.

J. Henderson, O. Lemon, and K. Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*.

T. Paek and D. Chickering. 2005. The markov assumption in spoken dialogue management. In *6th SIGDial Workshop on Discourse and Dialogue*.

S. Singh, M. Kearns, D. Litman, and M. Walker. 1999. Reinforcement learning for spoken dialogue systems. In *Proc. NIPS '99*.

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. Optimizing dialogue managment with reinforcement learning: Experiments with the njfun system. *JAIR*, 16.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. The MIT Press.

M. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *JAIR*, 12.

J. Williams and S. Young. 2003. Using wizard-of-oz simulations to bootstrap reinforcement learning-based dialog management systems. In *4th SIGdial Workshop on Discourse and Dialogue*.

C. Zinn, J. Moore, and M. Core. 2005. Intelligent information presentation for tutoring systems. *Intelligent Information Presentation*.