Comparing User Simulation Models For Dialog Strategy Learning

Hua Ai
University of Pittsburgh
Intelligent Systems Program
Pittsburgh PA, 15260, USA
hua@cs.pitt.edu

Joel R. Tetreault
University of Pittsburgh
LRDC
Pittsburgh PA, 15260, USA
tetreaul@pitt.edu

Diane J. Litman
University of Pittsburgh
Dept. of Computer Science
LRDC
Pittsburgh PA, 15260, USA
litman@cs.pitt.edu

Abstract

This paper explores what kind of user simulation model is suitable for developing a training corpus for using Markov Decision Processes (**MDPs**) to automatically learn dialog strategies. Our results suggest that with sparse training data, a model that aims to randomly explore more dialog state spaces with certain constraints actually performs at the same or better than a more complex model that simulates realistic user behaviors in a statistical way.

1 Introduction

Recently, user simulation has been used in the development of spoken dialog systems. In contrast to experiments with human subjects, which are usually expensive and time consuming, user simulation generates a large corpus of user behaviors in a low-cost and time-efficient manner. For example, user simulation has been used in evaluation of spoken dialog systems (López-Cózar et al., 2003) and to learn dialog strategies (Scheffler, 2002). However, these studies do not systematically evaluate how helpful a user simulation is. (Schatzmann et al., 2005) propose a set of evaluation measures to assess the realness of the simulated corpora (i.e. how similar are the simulated behaviors and human behaviors). Nevertheless, how realistic a simulated corpus needs to be for different tasks is still an open question.

We hypothesize that for tasks like system evaluation, a more realistic simulated corpus is preferable. Since the system strategies are evaluated and adapted based on the analysis of these simulated dialog behaviors, we would expect that these behaviors are what we are going to see in the test phase when the systems interact with human users. However, for automatically learning dialog strategies, it is not clear how realistic versus how exploratory (Singh et al., 2002) the training corpus should be. A training corpus needs to be exploratory with respect to the chosen dialog system actions because if a certain action is never tried at certain states, we will not know the value of taking that action in that state. In (Singh et al., 2002), their system is designed to randomly choose one from the allowed actions with uniform probability in the training phase in order to explore possible dialog state spaces. In contrast,we use user simulation to generate exploratory training data because in the tutoring system we work with, reasonable tutor actions are largely restricted by student performance. If certain student actions do not appear, this system would not be able to explore a state space randomly .

This paper investigates what kind of user simulation is good for using Markov Decision Processes (**MDPs**) to learn dialog strategies. In this study, we compare three simulation models which differ in their efforts on modeling the dialog behaviors in a training corpus versus exploring a potentially larger dialog space. In addition, we look into the impact of different state space representations and different reward functions on the choice of simulation models.

2 System and Corpus

Our system is a speech-enabled Intelligent Tutoring System that helps students understand qualitative physics questions. The dialog policy was deterministic and hand-crafted in a finite state paradigm (Ai et al., 2006). We collected 130 dialogs (1019 student utterances) with 26 human subjects. Correctness (correct(**c**), incorrect(**ic**)) is automatically judged by the system[1] and kept in the system's logs. Percent incorrectness (**ic%**) is also automatically calculated and logged. Each student utterance was manually annotated for certainty (*certain*, *uncertain*, *neutral*, *mixed*) in a previous study[2] based on both lexical and prosodic information. In this study, we use a two-way classification (certain(**cert**), notcertain(**ncert**)), where we collapse *uncertain*, *neutral*, and *mixed* to be **ncert** to balance our data. An example of coded dialog between the tutor (**T**) and a student (**S**) is given in Table 1.

T1:	Which law of motion would you use?
S1:	Newton's second law? [**ic**, **ic%**=1, **ncert**]
T2:	Well... The best law to use is Newton's third law. Do you recall what it says?
S2:	For every action there is an equal and opposite reaction? [**c**, **ic%**=50%, **ncert**]

Table 1: Sample coded dialog excerpt.

3 Experimental Setup

3.1 Learning Task

Our current system can only respond to the correctness of a student's utterances; the system thus ignores other underlying information, for example, certainty which is believed to provide useful information for the tutor. In our corpus, the strength of the tutor's minimal feedback (defined below) is in fact strongly correlated with the percentage of student certainty (chi-square test, $p<0.01$). Strong Feedback (**SF**) is when the tutor clearly states whether the student's answer is correct or incorrect (i.e., "This is great!"); Weak Feedback (**WF**) is when the tutor does not comment on the correctness of a student's answer or gives slightly negative feedback such as "well". Our goal is to learn how to manipulate the strength of the tutor minimal feedback in order to maximize student's overall certainty in the entire dialog. We keep the other parts of the tutor feedback (e.g. explanations, questions) so the system's original design of maximizing the percentage of student correct answers is utilized.

3.2 Simulation Models

All three models we describe below are trained from the real corpus we collected. We simulate on the word level because generating student's dialog acts alone does not provide sufficient information for our tutoring system to decide the next system's action. Thus, the output of the three models is a student utterance along with the student certainty (**cert**, **ncert**). Since it is hard to generate a natural language utterance for each tutor's question, we use the student answers in the real corpus as the candidate answers for the simulated students (Ai et al., 2006). In addition, we simulate student certainty in a very simple way: the simulation models output the certainty originally associated with that utterance.

Probabilistic Model (PM) is meant to capture realistic student behavior in a probabilistic way. Given a certain tutor question along with a tutor feedback, it will first compute the probabilities of the four types of student answers from the training corpus: **c** and **cert**, **c** and **ncert**, **ic** and **cert**, and **ic** and **ncert**. Then, following this distribution, the model selects the type of student answers to output, and then it picks an utterance that satisfies the correctness and certainty constraints of the chosen answer type from the candidate answer set and outputs that utterance. We implement a back-off mechanism to count possible answers that do not appear in the real corpus.

Total Random Model (TRM) ignores what the current question is or what feedback is given. It randomly picks one utterance from all the utterances in the entire candidate answer set. This model tries to explore all the possible dialog states.

Restricted Random Model (RRM) differs from the **PM** in that given a certain tutor question and a tutor feedback, it chooses to give a **c** and **cert**, **c** and **ncert**, **ic** and **cert**, or **ic** and **ncert** answer with equal probability. This model is a compromise between the exploration of the dialog state space and the realness of generated user behaviors.

3.3 MDP Configuration

A MDP has four main components: states, actions, a policy, and a reward function. In this study, the actions allowed in each dialog state are **SF** and **WF**;

[1] Kappa of 0.79 is gained comparing to human judgements.

[2] Kappa of 0.68 is gained in a preliminary agreement study.

the policy we are trying to learn is in every state whether the tutor should give **SF** and **WF** in order to maximize the percent certainty in the dialog.

Since different state space representations and reward functions have a strong impact on the MDP policy learning, we investigate different configurations to avoid possible bias introduced by certain configurations. We use two state space representations: **SSR1** uses the correctness of current student turn and percent incorrectness so far; and **SSR2** adds in the certainty of the current student turn on top of **SSR1**. Two reward functions are investigated: in **RF1**, we assign +100 to every dialog that has a percent certainty higher than the median from the training corpus, and -100 to every dialog that has a percent certainty below the median; in **RF2**, we assign different rewards to every different dialog by multiplying the percent certainty in that dialog with 100. Other MDP parameter settings are the same as described in (Tetreault et al., 2006).

3.4 Methodology

We first let the three simulation models interact with the original system to generate different training corpora. Then, we learn three MDP policies in a fixed configuration from the three training corpora separately. For each configuration, we run the simulation models until we get enough training data such that the learned policies on that corpus do not change anymore (40,000 dialogs are generated by each model). After that, the learned new policies are implemented into the original system respectively [3]. Finally, we use our most realistic model, the **PM**, to interact with each new system 500 times to evaluate the new systems' performances. We use two evaluation measures. **EM1** is the number of dialogs that would be assigned +100 using the old median split. **EM2** is the average of percent certainty in every single dialog from the newly generated corpus. A policy is considered better if it can improve the percentage of certainty more than other policies, or has more dialogs that will be assigned +100. The baseline for **EM1** is 250, since half of the 500 dialogs would be assigned +100 using the old median split. The baseline for **EM2** is 35.21%, which is obtained by calculating the percent certainty in the corpus generated by the 40,000 interactions between the **PM** and the original system.

4 Results and Discussion

Table 2 summarizes our results. There are two columns under each "state representation+reward function" configuration, presenting the results using the two evaluation approaches. **EM1** measures exactly what **RF1** tries to optimize; while **EM2** measures exactly what **RF2** tries to optimize. However, we show the results evaluated by both **EM1** and **EM2** for all configurations since the two evaluation measures have their own practical values and can be deployed under different design requirements. All results that significantly[4] outperform the corresponding baselines are marked with $*$.

When evaluating using **EM1**, the **RRM** significantly[4] outperforms the other two models in all configurations (in **bold** in Table 2). Also, the **PM** performs better (but not statistically significantly) than the **TRM**. When evaluating on **EM2**, the **RRM** significantly[4] outperforms the other two when using **SSR1** and **RF1** (in **bold** in Table 2). In all other configurations, the three models do not differ significantly. It is not surprising that the **RRM** outperforms the **PM** in most of the cases even when we test on the **PM**. (Schatzmann et al., 2005) also observe that a good model can still perform well when tested on a poor model.

We suspect that the performance of the **PM** is harmed by the data sparsity issue in the real corpus that we trained the model on. Consider the case of **SSR1**: 25.8% of the potentially possible dialog states do not exist in the real corpus. Although we implement a back-off mechanism, the **PM** will still have much less chance to transition to the states that are not observed in the real corpus. Thus, when we learn the MDP policy from the corpus generated by this model, the actions to take in these less-likely states are not fully learned. In contrast, the **RRM** transitions from one state to each of the next possible states with equal probability, which compensates for the data sparsity problem. We further examine the results obtained using **SSR1** and **RF1** and evaluated

[3] For example, the policy learned from the training corpus generated by the **RRM** with **SSR1** and **RF1** is: give **SF** when the current student answer is **ic** and **ic%**>50%, otherwise give **WF**.

[4] Using 2-sided t-test with Bonferroni correction, $p<0.05$.

Model Name	SSR1+RF1		SSR2+RF1		SSR1+RF2		SSR2+RF2	
	EM1	EM2	EM1	EM2	EM1	EM2	EM1	EM2
Probabilistic Model	222	36.30%	217	37.63%	197	40.78%∗	197	40.01%∗
Total Random Model	192	36.30%	211	38.57%	188	40.21%∗	179	40.21%∗
Restricted Random Model	**390**∗	**46.11%**∗	**368**∗	37.27%	**309**	40.21%∗	**301**	40.21%∗

Table 2: Evaluation of the new policies trained with the three simulation models

by **EM1** to confirm our hypothesis. When looking into the frequent states[5], 70.1% of them are seen frequently in the training corpus generated by the **PM**, while 76.3% are seen frequently in the training corpus generated by the **RRM**. A higher percentage indicates the policy might be better trained with more training instances. This explains why the **RRM** outperforms the **PM** in this case.

While the **TRM** also tries to explore dialog state space, only 65.2% of the frequent states in testing phase are observed frequently in the training phase. This is because the Total Random Model answers 90% of the questions incorrectly and often goes deeply down the error-correction paths. It does explore some states that are at the end of the paths, but since these are the infrequent states in the test phase, exploring these states does not actually improve the model's performance much. On the other hand, while the student correctness rate in the real corpus is 60%, the **RRM** prevents itself from being trapped in the less-likely states on incorrect answer paths by keeping its correctness rate to be 50%.

Our results are preliminary but suggest interesting points in building simulation models: 1. When trained from a sparse data set, it may be better to use a **RRM** than a more realistic **PM** or a more exploratory **TRM**; 2. State space representation may not impact evaluation results as much as reward functions and evaluation measures, since when using **RF2** and evaluating with **EM2**, the differences we see using **RF1** or **EM1** become less significant.

In our future work, we are going to further investigate whether the trends shown in this paper generalize to on-line MDP policy learning. We also want to explore other user simulations that are designed for sparse training data (Henderson et al., 2005). More importantly, we are going to test the new policies with the other simulations and human subjects to validate the learning process.

Acknowledgements

NSF (0325054, 0328431) supports this research. The authors wish to thank Tomas Singliar for his valuable suggestions, Scott Silliman for his support on building the simulation system, and the anonymous reviewers for their insightful comments.

References

H. Ai and D. Litman. 2006. *Comparing Real-Real, Simulated-Simulated, and Simulated-Real Spoken Dialogue Corpora*. In Proc. AAAI Workshop on Statistical and Empirical Approaches for SDS.

J. Henderson, O.Lemon, and K.Georgila. 2005. *Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data.* In Proc. IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems.

R. López-Cózar, A. De la Torre, J. Segura, and A. Rubio. 2003. *Assessment of dialog systems by means of a new simulation technique*. Speech Communication (40): 387-407.

K. Scheffler. 2002. *Automatic Design of Spoken Dialog Systems.* Ph.D. diss., Cambridge University.

J. Schatzmann, K. Georgila, and S. Young. 2005. *Quantitative Evaluation of User Simulation Techniques for Spoken Dialog Systems.* In Proc. of 6th SIGdial.

J. Schatzmann, M. N. Stuttle, K. Weilhammer and S. Young. 2005. *Effects of the User Model on Simulation-based Learning of Dialogue Strategies.* In Proc. of ASRU05.

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. *Optimizing Dialog Managment with Reinforcement Learning: Experiments with the NJFun System.* Journal of Artificial Intelligence Research, (16):105-133.

J. Tetreault and D. Litman. 2006. *Comparing the Utility of State Features in Spoken Dialogue Using Reinforcement Learning.*. In Proc. NAACL06.

[5] We define frequent states to be those that comprise at least 1% of the entire corpus. These frequent states add up to more than 80% of the training/testing corpus. However, deciding the threshold of the frequent states in training/testing is an open question.