Setting Up User Action Probabilities in User Simulations for Dialog System Development

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

Diane Litman
University of Pittsburgh
Pittsburgh PA, 15260, USA
litman@cs.pitt.edu

Abstract

User simulations are shown to be useful in spoken dialog system development. Since most current user simulations deploy probability models to mimic human user behaviors, how to set up user action probabilities in these models is a key problem to solve. One generally used approach is to estimate these probabilities from human user data. However, when building a new dialog system, usually no data or only a small amount of data is available. In this study, we compare estimating user probabilities from a small user data set versus handcrafting the probabilities. We discuss the pros and cons of both solutions for different dialog system development tasks.

1 Introduction

User simulations are widely used in spoken dialog system development. Recent studies use user simulations to generate training corpora to learn dialog strategies automatically ((Williams and Young, 2007), (Lemon and Liu, 2007)), or to evaluate dialog system performance (López-Cózar et al., 2003). Most studies show that using user simulations significantly improves dialog system performance as well as speeds up system development. Since user simulation is such a useful tool, dialog system researchers have studied how to build user simulations from a variety of perspectives. Some studies look into the impact of training data on user simulations. For example, (Georgila et al., 2008) observe differences between simulated users trained from human users of different age groups. Other studies explore different simulation models, i.e. the mechanism of deciding the next user actions given the current dialog context. (Schatzmann et al., 2006) give a thorough review of different types of simulation models. Since most of these current user simulation techniques use probabilistic models to generate user actions, how to set up the probabilities in the simulations is another important problem to solve.

One general approach to set up user action probabilities is to learn the probabilities from a collected human user dialog corpus ((Schatzmann et al., 2007b), (Georgila et al., 2008)). While this approach takes advantage of observed user behaviors in predicting future user behaviors, it suffers from the problem of learning probabilities from one group of users while potentially using them with another group of users. The accuracy of the learned probabilities becomes more questionable when the collected human corpus is small. However, this is a common problem in building new dialog systems, when often no data[1] or only a small amount of data is available. An alternative approach is to handcraft user action probabilities ((Schatzmann et al., 2007a), (Janarthanam and Lemon, 2008)). This approach is less data-intensive, but requires nontrivial work by domain experts. What is more, as the number of probabilities increases, it is hard even for the experts to set the probabilities. Since both handcrafting and training user action probabilities have their own pros and cons, it is an interesting research question to investigate which approach is better for a certain task given the amount of data that is available.

In this study, we investigate a manual and a trained approach in setting up user action probabilities, applied to building the same probabilistic simulation model. For the manual user simulations, we look into two sets of handcrafted probabilities which use the same expert knowledge but differ in individual probability values. This aims to take into account small variations that can possi-

[1] When no human user data is collected with the dialog system, Wizard-of-Oz experiments can be conducted to collect training data for building user simulations.

bly be introduced by different domain experts. For the trained user simulations, we examine two sets of probabilities trained from user corpora of different sizes, since the amount of training data will impact the quality of the trained probability models. We compare the trained and the handcrafted simulations on three tasks. We observe that in our task settings, the two manual simulations do not differ significantly on any tasks. In addition, there is no significant difference among the trained and the manual simulations in generating corpus level dialog behaviors as well as in generating training corpora for learning dialog strategies. When comparing on a dialog system evaluation task, the simulation trained from more data significantly outperforms the two manual simulations, which again outperforms the simulation trained from less data.

Based on our observations, we answer the original question of how to design user action probabilities for simulations that are similar to ours in terms of the complexity of the simulations[2]. We suggest that handcrafted user simulations can perform reasonably well in building a new dialog system, especially when we are not sure that there is enough data for training simulation models. However, once we have a dialog system, it is useful to collect human user data in order to train a new user simulation model since the trained simulations perform better than the handcrafted user simulations on more tasks. Since how to decide whether enough data is available for simulation training is another research question to answer, we will further discuss the impact of our results later in Section 6.

2 Related Work

Most current simulation models are probabilistic models in which the models simulate user actions based on dialog context features (Schatzmann et al., 2006). We represent these models as:

$$P(\text{user action}|feature_1, \ldots, feature_n) \quad (1)$$

The number of probabilities involved in this model is:

$$(\text{\# of possible actions-1}) * \prod_{k=1}^{n} (\text{\# of feature values}). \quad (2)$$

Some studies handcraft these probabilities. For example, (Schatzmann et al., 2007a) condition the user actions on user's goals and the agenda to reach those goals. They manually author the probabilities in the user's agenda update model and the goal update model, and then calculate the user action probabilities based on the two models. (Janarthanam and Lemon, 2008) handcraft 15 probabilities in simulated users' initial profiles and then author rules to update these probabilities during the dialogs.

Other studies use a human user corpus as the training corpus to learn user action probabilities in user simulations. Since the human user corpus often does not include all possible actions that users may take during interactions with the dialog system, different strategies are used to account for user actions that do not appear in the training corpus but may be present when testing the user simulations. For example, (Schatzmann et al., 2007b) introduce a summary space approach to map the actual dialog context space into a more tractable summary space. Then, they use forward and backward learning algorithms to learn the probabilities from a corpus generated by 40 human users (160 dialogs). (Rieser and Lemon, 2006) use a two step approach in computing the probabilities from a corpus consisting of dialogs from 24 human users (70 dialogs). They first cluster dialog contexts based on selected features and then build conditional probability models for each cluster.

In our study, we build a conditional probability model which will be described in detail in Section 3.2.1. There are 40 probabilities to set up in this model[3]. We will explain different approaches to assign these probabilities later in Section 3.2.2.

3 System and User Simulations

In this section, we describe the dialog system, the human user corpus we collected with the system, and the user simulation we used.

3.1 System and Corpus

The ITSPOKE system (Litman and Silliman, 2004) is an Intelligent Tutoring System which teaches Newtonian physics. It is a speech-enhanced version of the Why2-Atlas tutoring system (Vanlehn et al., 2002). During the interaction with students, the system initiates a spoken tutoring dialog to correct misconceptions and to

[2]The number of user action probabilities and the simulated user behaviors will impact the design choice.

[3]There are 2 possible actions in our model, 20 possible values for the first feature **qCluster** and 2 possible values for the second feature **prevCorrectness** as described later in Section 3.2.1. Using Equation 2, 40=(2-1)*20*2.

SYSTEM1:	Do you recall what Newton's third law says? [**3rdLaw**]
Student1:	Force equals mass times acceleration. [**ic**, **c%=0**, **ncert**]
SYSTEM2:	Newton's third law says ... If you hit the wall harder, is the force of your fist acting on the wall greater or less? [**3rdLaw**]
Student2:	Greater. [**c**, **c%=50%**,**cert**]
	Dialog goes on

Table 1: Sample coded dialog excerpt.

elicit further explanation. A pretest is given before the interaction and a posttest is given afterwards. We calculate a Normalized Learning Gain for each student to evaluate the performance of the system in terms of the student's knowledge gain:

$$NLG = \frac{\text{posttest score - pretest score}}{\text{1-pretest score}} \quad (3)$$

The current tutoring dialog strategy was handcrafted in a finite state paradigm by domain experts, and the tutor's response is based only on the correctness of the student's answer[4]. However, tutoring research (Craig et al., 2004) suggests that other underlying information in student utterances (e.g., student certainty) is also useful in improving learning. Therefore, we are working on learning a dialog strategy to also take into account student certainty.

In our prior work, a corpus of 100 dialogs (1388 student turns) was collected between 20 human subjects (5 dialogs per subject) and the ITSPOKE system. Correctness (correct(**c**), incorrect(**ic**)) is automatically judged by the system and is kept in the system's logs. We also computed the student's correctness rate (**c%**) and labeled it after every student turn. Each student utterance was manually annotated for certainty (certain(**cert**), notcertain(**ncert**)) in a previous study based on both lexical and prosodic information[5]. In addition, we manually clustered tutor questions into 20 clusters based on the knowledge that is required to answer that question, e.g. questions on Newton's Third Law are put into a cluster labeled as (**3rdLaw**). There are other clusters such as **gravity**, **acceleration**, etc. An example of a coded dialog between the system and a student is given in Table 1.

[4]Despite the limitation of the current system, students learn significantly after interacting with the system.

[5]Kappa of 0.68 is gained in the agreement study.

3.2 User Simulation Model and Model Probabilities Set-up

3.2.1 User Simulation Model

We build a Knowledge Consistency Model[6] (KC Model) to simulate consistent student behaviors while interacting with a tutoring system. According to learning literature (Cen et al., 2006), once a student acquires certain knowledge, his/her performance on similar problems that require the same knowledge (i.e. questions from the same cluster we introduced in Section 3.1) will become stable. Therefore, in the KC Model, we condition the student action **stuAction** based on the cluster of tutor question (**qCluster**) and the student's correctness when last encountering a question from that cluster (**prevCorrectness**): $P(stuAction|qCluster, prevCorrectness)$. For example, in Table 1, when deciding the student's answer after the second tutor question, the simulation looks back into the dialog and finds out that the last time (in Student1) the student answered a question from the same cluster **3rdLaw** incorrectly. Therefore, this time the simulation gives a correct student answer based on the probability $P(c|3rdLaw, ic)$.

Since different groups of students often have different learning abilities, we examine such differences among our users by grouping the users based on Normalized Learning Gains (NLG), which is an important feature to describe user behaviors in tutoring systems. By dividing our human users into high/low learners based on the median of NLG, we find a significant difference in the NLG of the two groups based on 2-tailed t-tests ($p < 0.05$). Therefore, we construct a simulation to represent low learners and another simulation to represent high learners to better characterize the differences in high/low learners' behaviors. Similar approaches are adopted in other studies in building user simulations for dialog systems (e.g., (Georgila et al., 2008) simulate old versus young users separately).

Our simulation models work on the word level [7] because generating student dialog acts alone does not provide sufficient information for our tutoring system to decide the next system action. Since it is hard to generate a natural language utterance for each tutor's question, we use the student answers

[6]This is the best model we built in our previous studies (Ai and Litman, 2007).

[7]See (Ai and Litman, 2006) for more details.

in the human user corpus as the candidate answers for the simulated students.

3.2.2 Model Probabilities Set-up

Now we discuss how to set up user action probabilities in the KC Model. We compare learning probabilities from human user data to handcrafting probabilities based on expert knowledge. Since we represent high/low learners using different models, we build simulation models with separate user action probabilities to represent the two groups of learners.

When learning the probabilities in the Trained KC Models, we calculate user action probabilities for high/low learners in our human corpus separately. We use add-one smoothing to account for user actions that do not appear in the human user corpus. For the first time the student answers a question in a certain cluster, we back-off the user action probability to P(stuAction | average correctness rate of this question in human user corpus). We first train a KC model using the data from all 20 human users to build the TrainedMore (**Tmore**) Model. Then, in order to investigate the impact of the amount of training data on the quality of trained simulations, we randomly pick 5 out of the 10 high learners and 5 out of the 10 low learners to get an even smaller human user corpus. We train the TrainedLess (**Tless**) Model from this small corpus .

When handcrafting the probabilities in the Manual KC Models[8], the clusters of questions are first grouped into three difficulty groups (**Easy**, **Medium**, **Hard**). Based on expert knowledge, we assume on average 70% of students can correctly answer the tutor questions from the Easy group, while for the Medium group only 60% and for the hard group 50%. Then, we assign a correctness rate higher than the average for the high learners and a corresponding correctness rate lower than the average for the low learners. For the first Manual KC model (**M1**), within the same difficulty group, the same two probabilities $P_1(stuAction|qCluster_i, prevCorrectness = c)$ and $P_2(stuAction|qCluster_i, prevCorrectness = ic)$ are assigned to each $cluster_i$ as the averages for the corresponding high/low learners. Since a different human expert will possibly provide a slightly different set of probabilities even based on the same mechanism, we also design another set of probabilities to account for such variations. For the second Manual KC model (**M2**), we allow differences among the clusters within the same difficulty group. For the clusters in each difficulty group, we randomly assign a probability that differs no more than 5% from the average. For example, for the easy clusters, we assign average probabilities of high/low learners between [65%, 75%].

Although human experts may differ to some extent in assigning individual probability values, we hypothesize that in general a certain amount of expertise is required in assigning these probabilities. To investigate this, we build a baseline simulation with no expert knowledge, which is a Random Model (**Ran**) that randomly assigns values for these user action probabilities.

[8]The first author of the paper acts as the domain expert.

4 Evaluation Measures

In this section, we introduce the evaluation measures for comparing the simulated corpora generated by different simulation models to the human user corpus. In Section 4.1, we use a set of widely used domain independent features to compare the simulated and the human user corpora on corpus-level dialog behaviors. These comparisons give us a direct impression of how similar the simulated dialogs are to human user dialogs. Then, we compare the simulations in task-oriented contexts. Since simulated user corpora are often used as training corpora for using MDPs to learn new dialog strategies, in Section 4.2 we estimate how different the learned dialog strategies would be when trained from different simulated corpora. Another way to use user simulation is to test dialog systems. Therefore, in Section 4.3, we compare the user actions predicted by the various simulation models with actual human user actions.

4.1 Measures on Corpus Level Dialog Behaviors

We compare the dialog corpora generated by user simulations to our human user corpus using a comprehensive set of corpus level measures proposed by (Schatzmann et al., 2005). Here, we use a subset of the measures which describe high-level dialog features that are applicable to our data. The measures we use include the number of student turns (Sturn), the number of tutor turns (Tturn), the number of words per student turn (Swordrate), the number of words per tutor turn (Twordrate), the ratio of system/user words per dialog (WordRatio),

and the percentage of correct answers (cRate).

4.2 Measures on Dialog Strategy Learning

In this section, we introduce two measures to compare the simulations based on their performance on a dialog strategy learning task. In recent studies (e.g., (Janarthanam and Lemon, 2008)), user simulations are built to generate a large corpus to build MDPs in using Reinforcement Learning (RL) to learn new dialog strategies. When building an MDP from a training corpus[9], we compute the transition probabilities $P(s_{t+1}|s_t, a)$ (the probability of getting from state s_t to the next state s_{t+1} after taking action a), and the reward of this transition $R(s_t, a, s_{t+1})$. Then, the expected cumulative value (V-value) of a state s can be calculated using this recursive function:

$$V(s) = \sum_{s_{t+1}} P(s_{t+1}|s_t, a)[R(s_t, a, s_{t+1}) + \gamma V(s_{t+1})] \quad (4)$$

γ is a discount factor which ranges between 0 and 1.

For our evaluation, we first compare the transition probabilities calculated from all simulated corpora. The transition probabilities are only determined by the states and user actions presented by the training corpus, regardless of the rest of the MDP configuration. Since the MDP configuration has a big impact on the learned strategies, we want to first factor this impact out and estimate the differences in learned strategies that are brought in by the training corpora alone. As a second evaluation measure, we apply reinforcement learning to the MDP representing each simulated corpus separately to learn dialog strategies. We compare the Expected Cumulative Rewards (ECRs)(Williams and Young, 2007) of these dialog strategies, which show the expectation of the rewards we can obtain by applying the learned strategies.

The MDP learning task in our study is to maximize student certainty during tutoring dialogs. The dialog states are characterized using the correctness of the current student answer and the student correctness rate so far. We represent the correctness rate as a binary feature: **lc** if it is below the training corpus average and **hc** if it is above the average. The end of dialog reward is assigned to be +100 if the dialog has a percent certainty higher than the median from the training corpus and -100 otherwise. The action choice of the tutoring system is to give a strong (**s**) or weak (**w**) feedback. A strong feedback clearly indicates the correctness of the current student answer while the weak feedback does not. For example, the second system turn in Table 1 contains a weak feedback. If the system says "Your answer is incorrect" at the beginning of this turn, that would be a strong feedback. In order to simulate student certainty, we simply output the student certainty originally associated in each student utterance. Thus, the output of the KC Models here is a student utterance along with the student certainty (cert, ncert). In a previous study (Ai et al., 2007), we investigated the impact of different MDP configurations by comparing the ECRs of the learned dialog strategies. Here, we use one of the best-performing MDP configurations, but vary the simulated corpora that we train the dialog strategies on. Our goal is to see which user simulation performs better in generating a training corpus for dialog strategy learning.

[9]In this paper, we use off-line model-based RL (Paek, 2006) rather than learning an optimal strategy online during system-user interactions.

4.3 Measures on Dialog System Evaluation

In this section, we introduce two ways to compare human user actions with the actions predicted by the simulations. The aim of this comparison is to assess how accurately the simulations can replicate human user behaviors when encountering the same dialog situation. A simulated user that can accurately predict human user behaviors is needed to replace human users when evaluating dialog systems.

We randomly divide the human user dialog corpus into four parts: each part contains a balanced amount of high/low learner data. Then we perform four fold cross validation by always using 3 parts of the data as our training corpus for user simulations, and the remaining one part of the data as testing data to compare with simulated user actions. We always compare high human learners only with simulation models that represent high learners and low human learners only with simulation models that represent low learners. Comparisons are done on a turn by turn basis. Every time the human user takes an action in the dialogs in the testing data, the user simulations are used to predict an action based on related dialog information from the human user dialog. For a KC Model, the related dialog information includes qCluster and prevCorrectness . We first compare the simulation

predicted user actions directly with human user actions. We define simulation accuracy as:

$$Accuracy = \frac{\text{Correctly predicted human user actions}}{\text{Total number of human user actions}} \quad (5)$$

However, since our simulation model is a probabilistic model, the model will take an action stochastically after the same tutor turn. In other words, we need to take into account the probability for the simulation to predict the right human user action. If the simulation outputs the right action with a small probability, it is less likely that this simulation can correctly predict human user behaviors when generating a large dialog corpus. We consider a simulated action associated with a higher probability to be ranked higher than an action with a lower probability. Then, we use the reciprocal ranking from information retrieval tasks (Radev et al., 2002) to assess the simulation performance[10]. Mean Reciprocal Ranking is defined as:

$$MRR = \frac{1}{A}\sum_{k=1}^{A}\frac{1}{rank_i} \quad (6)$$

In Equation 6, A stands for the total number of human user actions, $rank_i$ stands for the ranking of the simulated action which matches the i-th human user action.

Table 2 shows an example of comparing simulated user actions with human user actions in the sample dialog in Table 1. In the first turn **Student1**, a simulation model has a 60% chance to output an incorrect answer and a 40% chance to output a correct answer while it actually outputs an incorrect answer. In this case, we consider the simulation ranks the actions in the order of: ic, c. Since the human user gives an incorrect answer at this time, the simulated action matches with this human user action and the reciprocal ranking is 1. However, in the turn **Student2**, the simulation's output does not match the human user action. This time, the correct simulated user action is ranked second. Therefore, the reciprocal ranking of this simulation action is 1/2.

We hypothesize that the measures introduced in this section have larger power in differentiating different simulated user behaviors since every simulated user action contributes to the comparison between different simulations. In contrast, the measures introduced in Section 4.1 and Section 4.2 have less differentiating power since they compare at the corpus level.

[10](Georgila et al., 2008) use Precision and Recall to capture similar information as our accuracy, and Expected Precision and Expected Recall to capture similar information as our reciprocal ranking.

5 Results

We let all user simulations interact with our dialog system, where each simulates 250 low learners and 250 high learners. In this section, we report the results of applying the evaluation measures we discuss in Section 4 on comparing simulated and human user corpora. When we talk about significant results in the statistics tests below, we always mean that the p-value of the test is ≤ 0.05.

5.1 Comparing on Corpus Level Dialog Behavior

Figure 1 shows the results of comparisons using domain independent high-level dialog features of our corpora. The x-axis shows the evaluation measures; the y-axis shows the mean for each corpus normalized to the mean of the human user corpus. Error bars show the standard deviations of the mean values. As we can see from the figure, the Random Model performs differently from the human and all the other simulated models. There is no difference in dialog behaviors among the human corpus, the trained and the manual simulated corpora.

In sum, both the Trained KC Models and the Manual KC Models can generate human-like high-level dialog behaviors while the Random Model cannot.

5.2 Comparing on Dialog Strategy Learning Task

Next, we compare the difference in dialog strategy learning when training on the simulated corpora using similar approaches in (Tetreault and Litman, 2008). Table 3 shows the transition probabilities starting from the state (c, lc). For example, the first cell shows in the Tmore corpus, the probability of starting from state (c, lc), getting a strong feedback, and transitioning into the same state is 24.82%. We calculate the same table for the other three states (c, hc), (ic, lc), and (ic, hc). Using paired-sample t-tests with bonferroni corrections, the only significant differences are observed between the random simulated corpus and each of the other simulated corpora.

i-th Turn	human	Simulation Model	Simulation Output	CorrectlyPredictedActions	ReciprocalRanking
Student1	ic	60% ic, 40% c	ic	1	1
Student2	c	70% ic, 30% c	ic	0	1/2
Average	/	/	/	(1+0)/2	(1+1/2)/2

Table 2: An Example of Comparing Simulated Actions with Human User Actions.

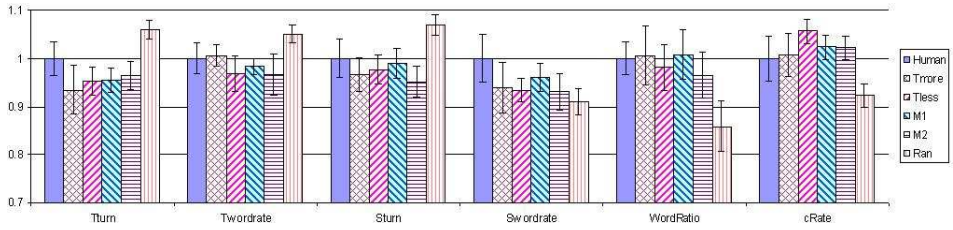

Figure 1: Comparison of human and simulated dialogs by high-level dialog features.

	Tmore	Tless	M1	M2	Ran
s→c_lc	24.82	31.42	25.64	22.70	13.25
w→c_lc	17.64	12.35	16.62	18.85	9.74
s→ic_lc	2.11	7.07	1.70	1.63	19.31
w→ic_lc	1.80	2.17	2.05	3.25	21.06
s→c_hc	29.95	26.46	22.23	31.04	10.54
w→c_hc	13.93	9.50	22.73	15.10	11.29
s→ic_hc	5.52	2.51	4.29	0.54	7.13
w→ic_hc	4.24	9.08	4.74	6.89	7.68

Table 3: Comparisons of MDP transition probabilities at state (c, lc) (Numbers in this table are percentages).

	Tmore	Tless	M1	M2	Ran
ECR	15.10	11.72	15.24	15.51	7.03
CI	±2.21	±1.95	±2.07	±3.46	±2.11

Table 4: Comparisons of ECR of learned dialog strategies.

We also use a MDP toolkit to learn dialog strategies from all the simulated corpora and then compute the Expected Cumulative Reward (ECR) for the learned strategies. In Table 4, the upper part of each cell shows the ECR of the learned dialog strategy; the lower part of the cell shows the 95% Confidence Interval (CI) of the ECR. We can see from the overlap of the confidence intervals that the only significant difference is observed between the dialog strategy trained from the random simulated corpus and the strategies trained from each of the other simulated corpora. Also, it is interesting to see that the CI of the two manual simulations overlap more with the CI of Tmore model than with the CI of the Tless model.

In sum, the manual user simulations work as well as the trained user simulation when being used to generate a training corpus to apply MDPs to learn new dialog strategies.

	Tmore	Tless	M1	M2	Ran
Accuracy	0.78 (±0.01)	0.60 (±0.02)	0.70 (±0.02)	0.72 (±0.02)	0.41 (±0.02)
MRR	0.72 (±0.02)	0.52 (±0.02)	0.63 (±0.02)	0.64 (±0.01)	0.32 (±0.02)

Table 5: Comparisons of correctly predicted human user actions.

5.3 Comparisons in Dialog System Evaluation

Finally, we compare how accurately the user simulations can predict human user actions given the same dialog context. Table 5 shows the averages and CIs (in parenthesis) from the four fold cross validations. The second row shows the results based on direct comparisons with human user actions, and the third row shows the mean reciprocal ranking of simulated actions. We observe that in terms of both the accuracy and the reciprocal ranking, the performance ranking from the highest to the lowest (with significant difference between adjacent ranks) is: the Tmore Model, both of the manual models (no significant differences between these two models), the Tless Model, and the Ran Model. Therefore, we suggest that the handcrafted user simulation is not sufficient to be used in evaluating dialog systems because it does not generate user actions that are as similar to human user actions. However, the handcrafted user simulation is still better than a user simulation trained with not enough training data. This result also indicates that this evaluation measure has more differentiating power than the previous measures since it captures significant differences that are not shown by the previous measures.

In sum, the Tmore simulation performs the best in predicting human user actions.

6 Conclusion and Future Work

Setting up user action probabilities in user simulation is a non-trivial task, especially when no training data or only a small amount of data is available. In this study, we compare several approaches in setting up user action probabilities for the same simulation model: training from all available human user data, training from half of the available data, two handcrafting approaches which use the same expert knowledge but differ slightly in individual probability assignments, and a baseline approach which randomly assigns all user action probabilities. We compare the built simulations from different aspects. We find that the two trained simulations and the two handcrafted simulations outperform the random simulation in all tasks. No significant difference is observed among the trained and the handcrafted simulations when comparing their generated corpora on corpus-level dialog features as well as when serving as the training corpora for learning dialog strategies. However, the simulation trained from all available human user data can predict human user actions more accurately than the handcrafted simulations, which again perform better than the model trained from half of the human user corpus. Nevertheless, no significant difference is observed between the two handcrafted simulations.

Our study takes a first step in comparing the choices of handcrafting versus training user simulations when only limited or even no training data is available, e.g., when constructing a new dialog system. As shown for our task setting, both types of user simulations can be used in generating training data for learning new dialog strategies. However, we observe (as in a prior study by (Schatzmann et al., 2007b)) that the simulation trained from more user data has a better chance to outperform the simulation trained from less training data. We also observe that a handcrafted user simulation with expert knowledge can reach the performance of the better trained simulation. However, a certain level of expert knowledge is needed in handcrafting user simulations since a random simulation does not perform well in any tasks. Therefore, our results suggest that if an expert is available for designing a user simulation when not enough user data is collected, it may be better to handcraft the user simulation than training the simulation from the small amount of human user data. However, it is another open research question to answer how much data is enough for training a user simulation, which depends on many factors such as the complexity of the user simulation model. When using simulations to test a dialog system, our results suggest that once we have enough human user data, it is better to use the data to train a new simulation to replace the handcrafted simulation.

In the future, we will conduct follow up studies to confirm our current findings since there are several factors that can impact our results. First of all, our current system mainly distinguishes the student answers as correct and incorrect. We are currently looking into dividing the incorrect student answers into more categories (such as partially correct answers, vague answers, or over-specific answers) which will increase the number of simulated user actions. Also, although the size of the human corpus which we build the trained user simulations from is comparable to other studies (e.g., (Rieser and Lemon, 2006), (Schatzmann et al., 2007b)), using a larger human corpus may improve the performance of the trained simulations. We are in the process of collecting another corpus which will consist of 60 human users (300 dialogs). We plan to re-train a simulation when this new corpus is available. Also, we would be able to train more complex models (e.g., a simulation model which takes into account a longer dialog history) with the extra data. Finally, although we add some noise into the current manual simulation designed by our domain expert to account for variations of expert knowledge, we would like to recruit another human expert to construct a new manual simulation to compare with the existing simulations. It would also be interesting to replicate our experiments on other dialog systems to see whether our observations will generalize. Our long term goal is to provide guidance of how to effectively build user simulations for different dialog system development tasks given limited resources.

Acknowledgments

The first author is supported by Mellon Fellowship from the University of Pittsburgh. This work is supported partially by NSF 0325054. We thank K. Forbes-Riley, P. Jordan and the anonymous reviewers for their insightful suggestions.

References

H. Ai and D. Litman. 2006. *Comparing Real-Real, Simulated-Simulated, and Simulated-Real Spoken*

Dialogue Corpora. In Proc. of the AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems.

H. Ai and D. Litman. 2007. *Knowledge Consistent User Simulations for Dialog Systems*. In Proc. of Interspeech 2007.

H. Ai, J. Tetreault, and D. Litman. 2007. *Comparing User Simulation Models for Dialog Strategy Learning*. In Proc. of NAACL-HLT 2007.

H. Cen, K. Koedinger and B. Junker. 2006. *Learning Factors Analysis-A General Method for Cognitive Model Evaluation and Improvement*. In Proc. of 8th International Conference on ITS.

S. Craig, A. Graesser, J. Sullins, and B. Gholson. 2004. *Affect and learning: an exploratory look into the role of affect in learning with AutoTutor*. Journal of Educational Media 29(3), 241250.

K. Georgila, J. Henderson, and O. Lemon. 2005. *Learning User Simulations for Information State Update Dialogue Systems*. In Proc. of Interspeech 2005.

K. Georgila, M. Wolters, and J. Moore. 2008. *Simulating the Behaviour of Older versus Younger Users when Interacting with Spoken Dialogue Systems*. In Proc. of 46th ACL.

S. Janarthanam and O. Lemon. 2008. *User simulations for online adaptation and knowledge-alignment in Troubleshooting dialogue systems*. In Proc. of the 12th SEMdial Workshop on on the Semantics and Pragmatics of Dialogues.

O. Lemon and X. Liu. 2007. *Dialogue Policy Learning for combinations of Noise and User Simulation: transfer results*. In Proc. of 8th SIGdial.

D. Litman and S. Silliman. 2004. *ITSPOKE: An Intelligent Tutoring Spoken Dialogue System*. In Companion Proc. of the Human Language Technology: NAACL.

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

T. Paek. 2006. *Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment*. In Proc. of Interspeech-06 Workshop on "Dialogue on Dialogues - Multidisciplinary Evaluation of Advanced Speech-based Interacive Systems".

D. Radev, H. Qi, H. Wu, and W. Fan. 2002. *Evaluating web-based question answering systems*. In Proc. of LREC 2002.

V. Rieser and O. Lemon. 2006. *Cluster-based User Simulations for Learning Dialogue Strategies*. In Proc. of Interspeech 2006.

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

J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. 2006. *A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies*. Knowledge Engineering Review 21(2): 97-126.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. 2007a. *Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System*. In Proc. of HLT/NAACL 2007.

J. Schatzmann, B. Thomson and S. Young. 2007b. *Statistical User Simulation with a Hidden Agenda*. In Proc. of 8th SIGdial.

J. Tetreault and D. Litman. 2008. *A Reinforcement Learning Approach to Evaluating State Representations in Spoken Dialogue Systems*. Speech Communication (Special Issue on Evaluating new methods and models for advanced speech-based interactive systems), 50(8-9): 683-696.

K. VanLehn, P. Jordan, C. Rosé, D. Bhembe, M. Böttner, A. Gaydos, M. Makatchev, U. Pappuswamy, M. Ringenberg, A. Roque, S. Siler, R. Srivastava, and R. Wilson. 2002. *The architecture of Why2-Atlas: A coach for qualitative physics essay writing*. In Proc. Intelligent Tutoring Systems Conference..

J. Williams and S. Young. 2007. *Partially Observable Markov Decision Processes for Spoken Dialog Systems*. Computer Speech and Language 21(2): 231-422.