User Simulation as Testing for Spoken Dialog Systems

Hua Ai*
Intelligent Systems Program
University of Pittsburgh
210 S. Bouquet St., Pittsburg, PA 15260
Hua@cs.pitt.edu

Fuliang Weng
Research and Technology Center
Robert Bosch LLC
4009 Miranda Ave., Palo Alto, CA 94304
Fuliang.weng@us.bosch.com

Abstract

We propose to use user simulation for testing during the development of a sophisticated dialog system. While the limited behaviors of the state-of-the-art user simulation may not cover important aspects in the dialog system testing, our proposed approach extends the functionality of the simulation so that it can be used at least for the early stage testing before the system reaches stable performance for evaluation involving human users. The proposed approach includes a set of evaluation measures that can be computed automatically from the interaction logs between the user simulator and the dialog system. We first validate these measures on human user dialogs using user satisfaction scores. We also build a regression model to estimate the user satisfaction scores using these evaluation measures. Then, we apply the evaluation measures on a simulated dialog corpus trained from the real user corpus. We show that the user satisfaction scores estimated from the simulated corpus are not statistically different from the real users' satisfaction scores.

1 Introduction

Spoken dialog systems are being widely used in daily life. The increasing demands of such systems require shorter system development cycles and better automatic system developing techniques. As a result, machine learning techniques are applied to learn dialog strategies automatically, such as reinforcement learning (Singh et al., 2002; Williams & Young, 2007), supervised learning (Henderson et al., 2005), etc. These techniques require a significant amount of training data for the automatic learners to sufficiently explore the vast space of possible dialog states and strategies. However, it is always hard to obtain training corpora that are large enough to ensure that the learned strategies are reliable. User simulation is an attempt to solve this problem by generating synthetic training corpora using computer simulated users. The simulated users are built to mimic real users' behaviors to some extent while allowing them to be programmed to explore unseen but still possible user behaviors. These simulated users can interact with the dialog systems to generate large amounts of training data in a low-cost and time-efficient manner. Many previous studies (Scheffler, 2002; Pietquin, 2004) have shown that the dialog strategies learned from the simulated training data outperform the hand-crafted strategies. There are also studies that use user simulation to train speech recognition and understanding components (Chung, 2004).

While user simulation is largely used in dialog system training, it has only been used in limited scope for testing specific dialog system components in the system evaluation phase (López-Cózar et al., 2003; Filisko and Seneff, 2006). This is partly because the state-of-the-art simulated users have quite limited abilities in mimicking human users' behaviors and typically over-generate possible dialog behaviors. This is not a major problem when using simulated dialog corpus as the training corpus for dialog strategy learning because the over-generated simulation behaviors would only provide the machine learners with a broader dialog state space to explore (Ai et al., 2007). However, realistic user behaviors are highly desired in the testing phase because the systems are evaluated and adjusted based on the analysis of the dialogs generated in this phase. Therefore, we would ex-

* This study was conducted when the author was an intern at Bosch RTC.

Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue, pages 164–171,
Columbus, June 2008.

pect that these user behaviors are what we will see in the final evaluation with human users. In this case, any over-generated dialog behaviors may cause the system to be blamed for untargeted functions. What is more, the simulated users cannot provide subjective user satisfaction feedback which is also important for improving the systems.

Since it is expensive and time-consuming to test every version of the system with a significant amount of paid subjects, the testing during the development is typically constrained to a limited number of users, and often, to repeated users who are colleagues or developers themselves. Thus, the system performance is not always optimized for the intended users.

Our ultimate goal is to supplement human testing with simulated users during the development to speed up the system development towards desired performance. This would be especially useful in the early development stage, since it would avoid conducting tests with human users when they may feel extremely frustrated due to the malfunction of the unstable system.

As a first attempt, we try to extend the state-of-the-art user simulation by incorporating a set of new but straightforward evaluation measures for automatically assessing the dialog system performance. These evaluation measures focus on three basic aspects of task-oriented dialog systems: understanding ability, efficiency, and the appropriateness of the system actions. They are first applied on a corpus generated between a dialog system and a group of human users to demonstrate the validity of these measures with the human users' satisfaction scores. Results show that these measures are significantly correlated with the human users' satisfactions. Then, a regression model is built to predict the user satisfaction scores using these evaluation measures. We also apply the regression model on a simulated dialog corpus trained from the above real user corpus, and show that the user satisfaction scores estimated from the simulated dialogs do not differ significantly from the real users' satisfaction scores. Finally, we conclude that these evaluation measures can be used to assess the system performance based on the estimated user satisfaction.

2 User Simulation Techniques

Most user simulation models are trained from dialog corpora generated by human users. Earlier models predict user actions based on simple relations between the system actions and the following user responses. (Eckert et al., 1997) first suggest a bigram model to predict the next user's action based on the previous system's action. (Levin et al., 2000) add constraints to the bigram model to accept the expected dialog acts only. However, their basic assumption of making the next user's action dependent only on the system's previous action is oversimplified. Later, many studies model more comprehensive user behaviors by adding user goals to constrain the user actions (Scheffler, 2002; Pietquin, 2004). These simulated users mimic real user behaviors in a statistical way, conditioning the user actions on the user goals and the dialog contexts. More recent research defines agenda for simulated users to complete a set of settled goals (Schatzmann et al., 2007). This type of simulated user updates the agenda and the current goal based on the changes of the dialog states.

In this study, we build a simulated user similar to (Schatzmann et al., 2007) in which the simulated user keeps a list of its goals and another agenda of actions to complete the goals. In our restaurant selection domain, the users' tasks are to find a desired restaurant based on several constraints specified by the task scenarios. We consider these restaurant constraints as the goals for the simulated user. At the beginning of the dialog, the simulated user randomly generates an agenda for the list of the ordered goals corresponding to the three constraints in requesting a restaurant. An agenda contains multiple ordered items, each of which consists of the number of constraints and the specific constraints to be included in each user utterance. During the dialog, the simulated user updates its list of goals by removing the constraints that have been understood by the system. It also removes from its agenda the unnecessary actions that are related to the already filled goals while adding new actions. New actions are added according to the last system's question (such as requesting the user to repeat the last utterance) as well as the simulated user's current goals. The actions that address the last system's question are given higher priorities then other actions in the agenda. For example, if the dialog system fails to understand the last user utterance and thus requests a clarification, the simulated user will satisfy the system's request before moving on to discuss a new constraint. The simulated user updated the agenda with the new actions after each user turn.

The current simulated user interacts with the system on the word level. It generates a string of words by instantiating its current action using predefined templates derived from previously collected corpora with real users. Random lexical errors are added to simulate a spoken language understanding performance with a word error rate of 15% and a semantic error rate of 11% based on previous experience (Weng et al., 2006).

3 System and Corpus

CHAT (Conversational Helper for Automotive Tasks) is a spoken dialog system that supports navigation, restaurant selection and mp3 player applications. The system is specifically designed for users to interact with devices and receive services while performing other cognitive demanding, or primary tasks such as driving (Weng et al., 2007). CHAT deploys a combination of off-the-shelf components, components used in previous language applications, and components specifically developed as part of this project. The core components of the system include a statistical language understanding (SLU) module with multiple understanding strategies for imperfect input, an information-state-update dialog manager (DM) that handles multiple dialog threads and mixed initiatives (Mirkovic and Cavedon, 2005), a knowledge manager (KM) that controls access to ontology-based domain knowledge, and a content optimizer that connects the DM and the KM for resolving ambiguities from the users' requests, regulating the amount of information to be presented to the user, as well as providing recommendations to users. In addition, we use Nuance 8.5[1] with dynamic grammars and classbased n-grams, for speech recognition, and Nuance Vocalizer 3.0 for text-to-speech synthesis (TTS). However, the two speech components, i.e., the recognizer and TTS are not used in the version of the system that interacts with the simulated users.

The CHAT system was tested for the navigation domain, the restaurant selection and the MP3 music player. In this study, we focus on the dialog corpus collected on the restaurant domain only. A small number of human users were used as dry-run tests for the system development from November, 2005 to January, 2006. We group the adjacent dry-runs to represent system improvement stages on a weekly basis. Table 1 shows the improvement stages, the dry-run dates which each stage includes, and the number of subjects tested in each stage. A final evaluation was conducted during January 19-31, 2006, without any further system modifications. This final evaluation involved 20 paid subjects who were recruited via internet advertisement.

Only the users in the final evaluation completed user satisfaction surveys after interacting with the system. In the survey, users were asked to rate the conversation from 6 perspectives, each on a 5-point scale: whether the system was easy to use, whether the system understood the user well, whether the interaction pattern was natural, whether the system's actions were appropriate, whether the system acted as expected, and whether the user was willing to use the system on a regular base. A user satisfaction score was computed as the average of the 6 ratings.

Stage	Dry-run Dates	Users
1	11/21/05, 11/22/05	2
2	11/30/05, 12/1/05, 12/2/05	3
3	12/7/05, 12/8/05	2
4	12/13/05, 12/14/05, 12/15/05	5
5	12/19/05, 12/20/05, 12/21/05	4
6	12/27/05, 12/28/05	2
7	1/4/06, 1/5/06	2
8	1/10/06, 1/11/06, 1/13/06	4
9	1/16/06, 1/17/06	3

Table 1: Dry-runs

Nine tasks of restaurant selections were used in both dry-runs and the final evaluation using 12 constraints in total (e.g., cuisine type, price level, location). These 12 constraints are spread across the nine tasks evenly with three constraints per task. In addition, each task is carefully worded based on the task-constrained and language-unconstrained guideline. In other words, we want the users to form an intended mental context while trying to prevent them from copying the exact phrasing in the task description. During the dry-runs, the users randomly pick three to four tasks to

[1] See http://www.nuance.com for details.

Utterances	Constraints Understood by the System	Action Types & Understanding Status
User1: I'm looking for a five star Thai [Italian] restaurant accept credit cards		Action: req_info Slots conveyed: 3 slots
System1: There are 7 restaurants that have a five star rating, accept credit cards, and serve Thai food in my database. Here are the first few: TeJUS, Slam Corner Cate, Setha's Restaurant. Should I continue?	service level=five star cuisine type=Thai payment type=credit card	Action: pro_info Slots accepted: 3 slots Completely matched: 2 slots
User2: I need to look for Italian restaurant		Action: rereq_info Slot conveyed: 1 slot
System2: Are you looking for a restaurant that serves Italian food?	service level=five star cuisine type=Italian (unconfirmed) payment type=credit card	Action: exp_confirm Slot updated: 1 slot Completely matched: 2.5 slot
User3: yes		Action: exp_confirm Slot conveyed: 1 slot
System3: I found 38 restaurants that serve Italian food, accept credit cards and have a five star rating, for example, Toto Café …	service level=five star cuisine type=Italian (confirmed) payment type=credit card	Action: pro_info Slots accepted: 3 slots Completely matched: 3 slots
User4: I'm done.		Action: close_dialog

Table 2: A Sample Dialog with Annotations.

test the system, while in the final evaluation each user is required to complete all of the 9 tasks. As a result of the final evaluation in the restaurant domain with 2500 restaurants, we reached a task completion rate of 94% with a word recognition rate of 85%, and a semantic accuracy rate of 89%.

4 Evaluation Measures

In this section, we describe in detail the evaluation measures covering three basic aspects of task-oriented dialog systems: understanding ability, efficiency, and the appropriateness of the system actions.

4.1 Understanding Ability Measures

Human-human dialog is a process to reach mutual understandings between the dialog partners by exchanging information through the dialog. This information exchanging process also takes place in the interaction between users and spoken dialog systems. In a task-oriented conversation, the dialog system's major task is to understand the users' needs in order to provide the right service. In the information-state update framework, the system continuously updates its information-states during the dialog while the users are conveying their requirements. If a misunderstanding occurs, there would be a mismatch between the users' requirements and the system's understandings. Thus, the error recovery dialog is needed to fix the mismatches. The error recovery dialog can be initiated either by the system by asking the user to rephrase or to repeat the previous utterance, or by the user to restate the previous request.

We use the percent of agreement between the system's and the user's understandings (**understandingAgreement**) to measure how well the system understands the user. The computation of this measure is illustrated through the example dialog in Table 2. In this table, the first column shows the system utterances and the user utterances received by the system. The correct words are shown in square brackets immediately after the misunderstood words (E.g., in Utterance "User1"). The second column represents semantic content from the users' utterances in the form of constraint-value pairs based on the system's understandings. This information can be automatically retrieved from the system logs. The third column includes the action types of the current system/user utterances. Since the dialog manager is an information-updating dialog manager that manages information in the format of slots, this column also shows the number of slots that are exchanged in the utterance and the number of matched slots. In our task domain, the user can request information (**req_info**), request the same information again (**rereq_info**), answer an explicit confirmation (**exp_confirm**), and close a dialog (**close_dialog**). The system can provide information (**pro_info**) or explicitly confirms (**exp_confirm**) the information. Another available system action that is not shown in this example is to ask the user to repeat/rephrase (**rephrase**), where the user can respond by providing the information again (**repro_info**).

In our experiment, we measure the understandings between the users and the system by comparing the values of the constraints that are specified by the users with their values understood by the system. In this dialog, the user specified all constraints in the first utterance:

Service level = Five star
Cuisine type = Italian
Payment type = Credit card

The first system utterance shows that the system understood two constraints but misunderstood the cuisine type, thus the percent agreement of mutual understandings is 2/3 at this time. Then, the user restated the cuisine type and the second system utterance confirmed this information. Since the system only asks for explicit information when its confidence is low, we count the system's understanding on the cuisine type as a 50% match with the user's. Therefore, the total percent agreement is 2.5/3. The user then confirmed that the system had correctly understood all constraints. Therefore, the system provided the restaurant information in the last utterance. The system's understanding matches 100% with the user's at this point.

The percent agreement of system/user understandings over the entire dialog is calculated by averaging the percent agreement after each turn. In this example, understandingAgreement is (2/3 + 2.5/3 + 1)/3 =83.3%. We hypothesize that the higher the understandingAgreement is, the better the system performs, and thus the more the user is satisfied. The matches of understandings can be calculated automatically from the user simulation and the system logs. However, since we work with human users' dialogs in the first part of this study, we manually annotated the semantic contents (e.g., cuisine name) in the real user corpus.

Previous studies (E.g., Walker et al., 1997) use a corpus level semantic accuracy measure (**semanticAccuracy**) to capture the system's understanding ability. SemanticAccuracy is defined in the standard way as the total number of correctly understood constraints divided by the total number of constraints mentioned in the entire dialog. The understandingAgreement measure we introduce here is essentially the averaged per-sentence semantic accuracy, which emphasizes the utterance level perception rather than a single corpus level average. The intuition behind this new measure is that it is better for the system to always understand something to keep a conversation going than for the system to understand really well sometimes but really bad at other times. We compute both measures in our experiments for comparison.

4.2 Efficiency Measure

Efficiency is another important measure of the system performance. A standard efficiency measure is the number of dialog turns. However, we would like to take into account the user's dialog strategy because how the user specifies the restaurant selection constraints has a certain impact on the dialog pace. Comparing two situations where one user specifies the three constraints of selecting a restaurant in three separate utterances, while another user specifies all the constraints in one utterance, we will find that the total number of dialog turns in the second situation is smaller assuming perfect understandings. Thus, we propose to use the ratio between the number of turns in the perfect understanding situation and the number of turns in practice (**efficiencyRatio**) to measure the system efficiency. The larger the efficiencyRatio is, the closer the actual number of turns is to the perfect understanding situation. In the example in Table 2, because the user chose to specify all the constraints in one utterance, the dialog length would be 2 turns in perfect understanding situation (excluding the last user turn which is always "I'm done"). However, the actual dialog length is 6 turns. Thus, the efficiencyRatio is 2/6.

Since our task scenarios always contain three constraints, we can calculate the length of the error-free dialogs based on the user's strategy. When the user specifies all constraints in the first utterance, the ideal dialog will have only 2 turns; when the user specifies two constraints in one utterance and the other constraints in a separate utterance, the ideal dialog will have 4 turns; when the user specifies all constraints one by one, the ideal dialog will have 6 turns. Thus, in the simulation environment, the length of the ideal dialog can be calculated from the simulated users' agenda. Then, the efficiencyRatio can be calculated automatically. We manually computed this measure for the real users' dialogs.

Similarly, in order to compare with previous studies, we also investigate the total number of dialog turns (**dialogTurns**) proposed as the efficiency measure (E.g., Möller et al., 2007).

4.3 Action Appropriateness Measure

This measure aims to evaluate the appropriateness of the system actions. The definition of appropriateness can vary on different tasks and different system design requirements. For example, some systems always ask users to explicitly confirm their utterances due to high security needs. In this case, an explicit confirmation after each user utterance is an appropriate system action. However, in other cases, frequent explicit confirmations may be considered as inappropriate because they may irritate the users. In our task domain, we define the only inappropriate system action to be providing information based on misunderstood user requirements. In this situation, the system is not aware of its misunderstanding error. Instead of conducting an appropriate error-recovering dialog, the system provides wrong information to the user which we hypothesize will decrease the user's satisfaction.

We use the percentage of appropriate system actions out of the total number of system actions (**percentAppropriate**) to measure the appropriateness of system actions. In the example in Table 2, only the first system action is inappropriate in all 3 system actions. Thus, the percent system action appropriateness is 2/3. Since we can detect the system's misunderstanding and the system's action in the simulated dialog environment, this measure can be calculated automatically for the simulated dialogs. For the real user corpus, we manually coded the inappropriate system utterances.

Note that the definition of appropriate action we use here is fairly loose. This is partly due to the simplicity of our task domain and the limited possible system/user actions. Nevertheless, there is also an advantage of the loose definition: we do not bias towards one particular dialog strategy since our goal here is to find some general and easily measurable system performance factors that are correlated with the user satisfaction.

5 Investigating Evaluation Measures on the Real User Corpus

In this section, we first validate the proposed measures using real users' satisfaction scores, and then show the differentiating power of these measures through the improvement curves plotted on the dry-run data.

5.1 Validating Evaluation Measures

To validate the evaluation measures introduced in Section 4, we use Pearson's correlation to examine how well these evaluation measures can predict the user satisfaction scores. Here, we only look at the dialog corpus in final evaluation because only these users filled out the user satisfaction surveys. For each user, we compute the average value of the evaluation measures across all dialogs generated by that user.

Evaluation Measure	Correlation	P-value
understandingAgreement	0.354	0.05
semanticAccuracy	*0.304*	*0.08*
efficiencyRatio	0.406	0.02
dialogTurns	*-0.321*	*0.05*
percentAppropriate	0.454	0.01

Table3: Correlations with User Satisfaction Scores.

Table 3 lists the correlation between the evaluation measures and the user satisfaction scores, as well as the p-value for each correlation. The correlation describes a linear relationship between these measures and the user satisfaction scores. For the measures that describe the system's understanding abilities and the measures that describe the system's efficiency, our newly proposed measures show higher correlations with the user satisfaction scores than their counterparts. Therefore, in the rest of the study, we drop the two measures used by the previous studies, i.e., semanticAccuracy and dialogTurns.

We observe that the user satisfaction scores are significantly positively correlated with all the three proposed measures. These correlations confirms our expectations: user satisfaction is higher when the system's understanding matches better with the users' requirements; when the dialog efficiency is closer to the situation of perfect understanding; or when the system's actions are mostly appropriate. We suggest that these measures can serve as indicators for user satisfaction.

We further use all the measures to build a regression model to predict the user satisfaction score. The prediction model is:

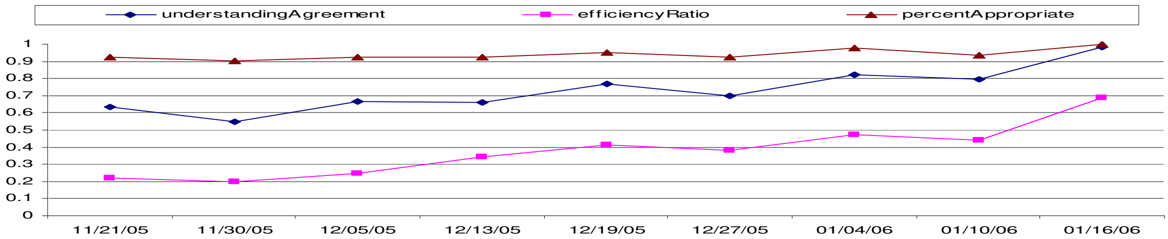

Figure 1: The Improvement Curves on Dry-run Data

User Satisfaction
= 6.123*percentAppropriate
+2.854*efficiencyRatio --- (1)
+0.864*understandingAgreement - 4.67

The R-square is 0.655, which indicates that 65.5% of the user satisfaction scores can be explained by this model. While this prediction model has much room for improvement, we suggest that it can be used to estimate the users' satisfaction scores for simulated users in the early system testing stage to quickly assess the system's performance. Since the weights are tuned based on the data from this specific application, the prediction model may not be used directly for other domains.

5.2 Assessing the Differentiating Power of the Evaluation Measures

Since this set of evaluation measures intends to evaluate the system's performance in the development stage, we would like the measures to be able to reflect small changes made in the system and to indicate whether these changes show the right trend of increased user satisfaction in reality. A set of good evaluation measures should be sensible to subtle system changes.

We assess the differentiating power of the evaluation measures using the dialog corpus collected during the dry-runs. The system was tested on a weekly basis as explained in Table 1. For each improvement stage, we compute the values for the three evaluation measures averaging across all dialogs from all users. Figure 1 shows the three improvement curves based on these three measures. The x-axis shows the first date of each improvement stage; the y-axis shows the value of the evaluation measures. We observe that all three curves show the right trends that indicate the system's improvements over the development stages.

6 Applying the Evaluation Measures on the Simulated Corpus

We train a goal and agenda driven user simulation model from the final evaluation dialog corpus with the real users. The simulation model interacts with the dialog system 20 times (each time the simulation model represents a different simulated user), generating nine dialogs on all of the nine tasks each time. In each interaction, the simulated users generate their agenda randomly based on a uniform distribution. The simulated corpus consists of 180 dialogs from 20 simulated users, which is of the same size as the real user corpus. The values of the evaluation measures are computed automatically at the end of each simulated dialog.

We compute the estimated user satisfaction score using Equation 1 for each simulated user. We then compare the user satisfaction scores of the 20 simulated users with the satisfaction scores of the 20 real users. The average and the standard deviation of the user satisfaction scores for real users are (3.79, 0.72), and the ones for simulated users are (3.77, 1.34). Using two-tailed t-test at significance level $p<0.05$, we observe that there are no statistically significant differences between the two pools of scores. Therefore, we suggest that the user satisfaction estimated from the simulated dialog corpus can be used to assess the system performance. However, these average scores only offer us one perspective in comparing the real with the simulated user satisfaction. In the future, we would like to look further into the differences between the distributions of these user satisfaction scores.

7 Conclusions and Future Work

User simulation has been increasingly used in generating large corpora for using machine learning techniques to automate dialog system design. However, user simulation has not been used much in testing dialog systems. There are two major concerns: 1. we are not sure how well the state-of-the-art user simulation can mimic realistic user behaviors; 2. we do not get important feedback on user satisfaction when replacing human users with simulated users. In this study, we suggest that while the simulated users might not be mature to use in the final system evaluation stage, they can be used in the early testing stages of the system development cycle to make sure that the system is functioning in the desired way. We further propose a set of evaluation measures that can be extracted from the simulation logs to assess the system performance. We validate these evaluation measures on human user dialogs and examine the differentiating power of these measures. We suggest that these measures can be used to guide the development of the system towards improving user satisfaction. We also apply the evaluation measures on a simulation corpus trained from the real user dialogs. We show that the user satisfaction scores estimated on the simulated dialogs do not significantly differ statistically from the real users' satisfaction scores. Therefore, we suggest that the estimated user satisfaction can be used to assess the system performance while testing with simulated users.

In the future, we would like to confirm our proposed evaluation measures by testing them on dialog systems that allows more complicated dialog structures and systems on other domains.

Acknowledgments

The authors would like to thank Zhongchao Fei, Zhe Feng, Junkuo Cao, and Baoshi Yan for their help during the simulation system development and the three anonymous reviewers for their insightful suggestions. All the remaining errors are ours.

References

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

G. Chung. 2004. *Developing a Flexible Spoken Dialog System Using Simulation*. In Proc. of ACL 04.

W. Eckert, E. Levin, and R. Pieraccini. 1997. *User Modeling for Spoken Dialogue System Evaluation*. In Proc. of IEEE workshop on ASRU.

E. Filisko and S. Seneff. 2006. *Learning Decision Models in Spoken Dialogue Systems Via User Simulation.* In Proc. of AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems.

J. Henderson, O. Lemon, and K. Georgila. 2005. *Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data*. In IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems.

E. Levin, R. Pieraccini, and W. Eckert. 2000. *A Stochastic Model of Human-Machine Interaction For learning Dialogue Strategies.* IEEE Trans. On Speech and Audio Processing, 8(1):11-23.

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.

D. Mirkovic and L. Cavedon. 2005. *Practical multi-domain, multi-device dialogue management,* PACLING'05: 6th Meeting of the Pacific Association for Computational Linguistics.

Sebastian Möller, Jan Krebber and Paula Smeele. 2006. *Evaluating the speech output component of a smart-home system*. Speech Communication (48): 1-27.

O. Pietquin, O. 2004. *A Framework for Unsupervised Learning of Dialog Strategies*. Ph.D. diss., Faculte Polytechnique de Mons.

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

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. *Optimizing DialogueManagement with Reinforcement Learning: Experiments with the NJFun System.* Journal of Artificial Intelligence Research (JAIR), vol. 16.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and Young. S. 2007. *Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System.* In Proc. of NAACL-HLT (short paper session).

F. Weng, S. Varges, B. Raghunathan, F. Ratiu, H. Pon-Barry, B. Lathrop, Q. Zhang, H. Bratt, T. Scheideck, R. Mishra, K. Xu, M. Purvey, A. Lien, M. Raya, S. Peters, Y. Meng, J. Russell, L. Cavedon, E. Shriberg, and H. Schmidt. 2006. *CHAT: A Conversational Helper for Automotive Tasks.* In Proc. of Interspeech.

F. Weng, B. Yan, Z. Feng, F. Ratiu, M. Raya, B. Lathrop, A. Lien, S. Varges, R. Mishra, F. Lin, M. Purver, H. Bratt, Y. Meng, S. Peters, T. Scheideck, B. Raghunathan and Z. Zhang. 2007. *CHAT to your destination.* In Proc. Of 8th SIGdial workshop on Discourse and Dialogue.

J. Williams and S. Young. 2006. *Partially Observable Markov Decision Processes for Spoken Dialog Systems*. Computer Speech and Language.

M. Walker, D. Litman, C. Kamm, and A. Abella. 1997. *PARADISE: A Framework for Evaluating Spoken Dialogue Agents*. In Proceedings of the 35th ACL.