# Reinforcement Learning for Spoken Dialogue Systems: Comparing Strengths and Weaknesses for Practical Deployment

*Tim Paek*

Microsoft Research
One Microsoft Way, Redmond, WA 98052
timpaek@microsoft.com

## Abstract

In a spoken dialogue system, the function of a dialogue manager is to select actions based on observed events and inferred beliefs. To formalize and optimize the action selection process, researchers have turned to reinforcement learning methods which represent the dynamics of a spoken dialogue as a fully or partially observable Markov Decision Process. Once represented as such, optimal policies prescribing what actions the system should take in order to maximize a reward function can be learned from data. Formerly, this task was assigned to the application developer, who typically hand-crafted rules or heuristics. In this position paper, we assess to what extent the action selection process can be automated by current state-of-the-art reinforcement learning methods for dialogue management. In examining the strengths and weaknesses of these methods with respect to practical deployment, we discuss the challenges that need to be overcome before these methods can become commonplace in deployed systems.

## 1. Introduction

In a spoken dialogue system (SDS), the function of a dialogue manager is to select actions based on observed events and inferred beliefs. Because these actions directly affect the usability — and ultimately, the task success of the SDS — researchers have turned to reinforcement learning to formalize and optimize the action selection process. One approach that has been gaining momentum is to represent the dynamics of a spoken dialogue as a fully or partially observable Markov Decision Process (MDP) and to derive an optimal policy that prescribes what actions the system should take in various states of the dialogue so as to maximize a reward function. Formerly, this task was assigned to the application developer, who typically hand-crafted rules or heuristics based on engineering experience, at best, and intuition, at worst. Authoring dialogue management strategies was not only time-consuming, but also prone to human error. With reinforcement learning, application developers could now exploit methods for automatically generating optimal strategies from data. This would render the task an optimization problem, which, once solved, could remove the "art" from the process and facilitate rapid application development. In this position paper, we consider to what extent the action selection process can indeed be automated by current state-of-the-art reinforcement learning methods.

This paper divides into three parts. In the first part, we provide relevant background on reinforcement learning methods that utilize a fully or partially observable MDP for dialogue management. In the second part, we consider the strengths and weaknesses of these methods with respect to practical deployment in non-research settings. In so doing, we discuss the challenges that need to be overcome before reinforcement learning methods can become commonplace in deployed systems, as well as opportunities for research. Finally, in the third part, we attempt to weigh the strengths against the weaknesses with respect to practical deployment and arrive at a summary judgment.

## 2. Background

Reinforcement learning addresses the problem of how an agent should act in dynamic environments so as to maximize a reward function [1]. Dialogue management can be construed as a reinforcement learning problem in that a SDS needs to take sequential actions, and those actions should be "optimal" in some way, such as maximizing a reward function. A central debate in the reinforcement learning literature concerns the use of models. *Model-free* approaches do not explicitly represent the dynamics of dialogue, but instead directly approximate a value function that measures the desirability of each environment state. These approaches offer near-optimal solutions that depend on systematic exploration of all actions in all states [2]. On the other hand, *model-based* approaches explicitly represent a model of the dynamics of dialogue to compute an estimate of the expected value of each action. With a model, the SDS can reduce the number of steps to learn a policy by simulating the effects of its actions at various states [1]. Perhaps for this reason, and for the fact that it is possible to derive a policy that is optimal with respect to the data, dialogue researchers have by and large pursued model-based reinforcement learning methods (see e.g., [3, 4]).

The framework underlying model-based reinforcement learning is that of the MDP, which can be characterized by a tuple $(S, A, T, R)$, where:

- $S$ is a finite set of states;
- $A$ is a finite set of actions;
- $T$ is a state-transition function such that $T(s', a, s) = p(s'|s, a)$; and
- $R : S \times A \mapsto \Re$ is a local reward function.

The objective of the SDS is to maximize its expected cumulative reward, which for the infinite-horizon, can include a discount factor to ensure that rewards accrued later are counted less than those accrued earlier:

$$E\left(\sum_{t=0}^{\infty} \gamma^t R_t\right) \quad (1)$$

where $\gamma$ is a geometric discount factor, $0 \leq \gamma < 1$. The discount factor encourages shorter dialogues and can be used to model pro-

cesses that can terminate at any time with probability 1 - $\gamma$, such as a user hanging up.

Unfortunately, an MDP requires complete knowledge of $S$, which may be intractably large if $S$ encodes all relevant dialogue history variables [5]. Furthermore, keeping track of *unobservable* states such as the user's intentions and beliefs, which can be inferred from observations such as the user's utterance, has been shown to improve performance [6, 7, 8]. If a SDS cannot observe all states $s \in S$, then the MDP is considered a Partially Observable MDP (POMDP), and can be characterized by the tuple $(S, A, T, R, O, Z)$, where:

- $S$, $A$, $T$, $R$ constitute an MDP;
- $O$ is a finite set of observations; and
- $Z$ is the observation function such that $Z(o, s, a) = p(o|s, a)$.

Because the dialogue system never knows with certainty the current state, it maintains a belief state $b(s)$, or a probability distribution over $S$. The local reward is then computed as the expected reward $\rho$ over belief states:

$$\rho(b, a) = \sum_{s \in S} R(s, a) \cdot b(s) \quad (2)$$

And the objective of the dialogue system is again to maximize its expected cumulative reward, as in equation 1.

Once a spoken dialogue has been formalized as above, a number of algorithms can be exploited to learn an optimal or near-optimal policy from data [9], where an optimal policy $\pi : S \mapsto A$ is a mapping from states to actions. With a POMDP, deriving a policy is more complicated (see [10] for a survey) as the policy $\pi$ becomes a mapping from initial beliefs and histories of actions and observations experienced so far — that is, $h_t = \langle a_0, o_1, a_1, o_2, ..., a_{t-1}, o_t \rangle$ — to actions. A POMDP policy can be represented in several ways. Perhaps the most pertinent for dialogue management is that of a finite-state controller, which can be learned when the optimal value function is piecewise linear and convex [11]. Given that some application developers may already be familiar with this kind of representation, it has been investigated for dialogue management [7].

# 3. Strengths and Weaknesses

The formalization in Section 2 identifies the key concepts for utilizing a fully or partially observable MDP for dialogue management. In this Section, we consider the strengths and weaknesses of these concepts with respect to practical deployment, and discuss the challenges that need to be overcome.

## 3.1. Objective Function

The appeal of reinforcement learning for speech research may be because dialogue management is cast into the same kind of statistical framework for optimization as speech recognition and spoken language understanding [5]. Unfortunately, whereas speech recognition and spoken language understanding is generally based on a maximum likelihood approach that essentially minimizes word or concept error rates, in dialogue management, the objective function is less clear. Equation 1 states that the SDS should maximize its expected cumulative reward. However, that objective function could also be based on post-hoc measures such as usability scores [4, 12], and construed to reflect whatever qualities application developers may want the SDS to possess.

The usual practice is to accept the expected cumulative reward, equation 1, as the objective function, and adjust the reward function $R$ to modify system behavior. However, the implications of an objective function for modeling dialogue have not been well investigated. First, it is unclear whether every dialogue can be viewed as an optimization problem with a specifiable objective function. Moreover, it is unclear how the choice of the expected cumulative reward, as opposed to any other objective function, affects the types of dialogue interaction that can be optimized.

Given an explicit objective function, a promising avenue for research is to optimize speech recognition and/or spoken language understanding using the same objective function as dialogue management. Just as spoken language understanding does not require correctly recognizing all the words, taking appropriate actions, in certain contexts, may not require correctly identifying *all* the words and concepts; e.g., in more conversational settings where maintaining social credibility outweighs everything.

For practical deployment, requiring an objective function to be explicitly specified may be both a strength and a weakness. It can be a strength in that the objective function can serve as an evaluation metric for controlled experiments. For example, it can be used to measure the effect of adding or removing features from the SDS. On the other hand, it can be a weakness in that most application developers have little to no experience with optimization or even statistics, and would likely be hard-pressed to specify an objective function. They may opt with the default setting, not understanding how it governs dialogue management, and later be puzzled as to why the SDS behaves as it does.

## 3.2. Reward Function

In pursuing the expected cumulative reward in equation 1, a local reward function $R$ must be specified. The typical practice is to assign a small negative reward for each dialogue turn and a large positive or negative reward upon completing the interaction successfully or unsuccessfully.

The local reward function is perhaps the most hand-crafted aspect of the reinforcement learning framework for dialogue management. The overall behavior of the system, as dictated by the policy, is very sensitive to changes in $R$, yet $R$ is almost always set by intuition, not data. For practical deployment, application developers may find it too difficult to assign $R$, as that entails having a good understanding about how the relative values of $R(s, a)$ for particular states and actions influence each other as well as the overall behavior of the system. Although application developers may be fine, for the most part, to go with reasonable defaults, if they are ever asked to modify the SDS so that it behaves in a certain way, this will be hard to do without conceptually understanding $R$ very well. They may be better off coding heuristics they understand than to try to tweak $R$.

Another problem is that $R$ is typically set so that it is static. However, it may be beneficial to have $R$ adapt to the user type and/or goal [13]. For example, for airline ticket reservation, without knowing anything about the user, a SDS may initially take actions that minimize penalties for turns. When it becomes clear that the user is more interested in exploring prices for vacations than purchasing a ticket, it may be worthwhile to increase the reward for engaging in longer dialogues and decrease the penalties for not being in a termination state. Of course, $R$ could just be construed to be a function of user type or goal. Depending on the

size of the state space $S$, that may or may not be feasible. Alternatively, different policies could be learned for different user types and alternated based upon the likelihood of a different user type. This kind of approach is similar to those that have been taken with adapting system and mixed initiative [14].

Finally, a promising avenue for future research is to learn the local reward function by watching a system behave according to its optimal policy and inferring $R$. This line of research is called "inverse reinforcement learning" [15]. For dialogue management, it may be possible to learn $R$ by observing human interlocutors engage in dialogue, allowing the SDS to mimic the human agent.

### 3.3. State Space and Transition Function

So far, the discussion has focused on local and cumulative rewards, but $R$ is a function of $S$. Modeling the state space $S$ is the most fundamental aspect of reinforcement learning, as it affects how the dynamics of the SDS operate, how rewards are assigned, and how tractable policy learning will be. For systems that currently utilize reinforcement learning, researchers have limited the state space to just a handful of variables, which constrains the kinds of domains and interactions that can be handled by the SDS. Furthermore, because $T$ is Markovian, state space variables must be selected so as to support the Markov assumption, which may not always be the best option [16]. To deal with scalability, researchers have exploited factored representations of $T$ to reduce the number of parameters that need to be estimated [7, 17] and introduced methods to scale POMDP dialogue managers to slot-filling problems of realistic size [18]. However, the state space still needs to be delineated up front. This is for the most part a manual task. One exception is research in learning the structure of the state space automatically from a myriad of candidate variables using Bayesian model selection techniques for factored representations [17]. But again, selecting candidate variables requires manual selection.

For practical deployment, it is unclear that application developers should accept whatever $S$ researchers utilize in their systems. First of all, the research community has not established best practices for modeling $S$, nor agreed upon a domain-independent set of variables that could be utilized in any SDS — which, by the way, constitutes an interesting challenge for the field. In extending $S$ to new domains, application developers may find that they need to model domain-dependent variables to improve the performance of the SDS. Alternatively, after the system has been deployed, they may find that they need to add new state variables. Unfortunately, adding new variables is not a minor fix. The entire policy has to be re-learned. As noted above, modeling the state space is fundamental, and affects everything.

### 3.4. Policy

The ultimate product of utilizing reinforcement learning methods for dialogue management is a policy that is optimal with respect to the data. Suppose that somehow tractability ceased to be a limiting factor, and that an optimal policy could be learned for arbitrarily large state and action spaces. Even in this ideal situation, the question of how beneficial an optimal policy is for application developers still remains.

Consider the issue of representation. As mentioned before, the policy can be represented in various ways, but all ways prescribe an optimal action that the SDS should take. Although it might seem as if this is what developers want — namely, a black box which tells the system what to do — it fundamentally wrests control of the dialogue flow from their hands, something that developers generally resist, for good reason. Of all the black boxes in a SDS (and there could be several, such as the speech recognizer), the one that affects the users the most is the dialogue manager because that is where all system actions are decided. Because the business of application developers revolves around satisfying the needs of their customers, if their customers tell them, for example, that they tried the SDS and was puzzled about how the system took a particular action after having gone through a series of exchanges, the developer better know how to fix that one particular action. This kind of fix, which would be relatively straightforward with dialogue strategies explicitly written out in code as conditional statements, is much harder to implement within the reinforcement learning framework because everything is interwoven. To get an action to change, and moreover, to change several turns into the dialogue, may entail modifying $R$, $S$, $T$, equation 1, and/or $\gamma$. In short, a fair amount of expertise in reinforcement learning is required to do the job.

To get a better idea of how resistant application developers may be to reinforcement learning, and statistical methods for dialogue management in general, consider the case of the statistical language model (SLM). If prevalence is any indication of preference, then context-free grammars are much more prevalent and preferred, in the commercial world than SLMs, despite the fact that SLMs have been around longer historically. Although application developers may be aware of the benefits of statistical modeling, they often prefer the deterministic character of context-free grammars because they know how to control and modify it. And when they modify it, they can predict exactly what the results will look like. This is not so clear with statistical dialogue management. Reinforcement learning strikes at something application developers want to maintain control of, at least as much as, if not more so, than language models. Worse, reinforcement learning has an even more complicated set of parameters to adjust in order to obtain the behavior they want.

Of course, the minute any change is made to an optimal policy, it ceases to be optimal with respect to the data. However, this may not be so bad if all that an application developer really wants is a first stab at a reasonable policy, to reduce design time. Furthermore, if the data was limited in some fashion, then it really does not matter if the policy is no longer optimal because it was only optimal with respect to the data on which it was learned anyway.

Online policy learning for dialogue management holds great promise in this regard. As mentioned in section 2, dialogue researchers have mostly focused on model-based reinforcement learning approaches. Although online policy learning algorithms exist for model-based approaches [19], model-free approaches are more commonly utilized [2]. Online policy learning is a promising area because the SDS would not be limited by the data on which it was trained. Without having explored all actions in all states, the system could engage in the type of exploration versus exploitation dilemma that characterizes classical reinforcement learning [1]. To date, very few dialogue researchers have investigated online policy learning [17].

Another promising avenue of research is the pursuit of domain-independent mechanisms for spoken dialogue, such as clarification strategies and error handling (see e.g., [20]). By separating out decisions or actions that application developers may not be interested in controlling, such as confirmations, it may be possible to design policies based upon state spaces and reward functions that are reusable.

### 3.5. Evaluation

The evaluation of reinforcement learning techniques for spoken dialogue systems has mostly centered on user simulation. Ever since researchers began examining reinforcement learning for dialogue management, they have realized that obtaining data to learn a policy would be problematic [21]. Because it is impractical, time-consuming and burdensome to have a SDS explore all different types of actions with real users, the idea was to learn a generative model of the user so that user actions could be simulated in response to system actions. With good user modeling, a SDS could be rapidly prototyped and evaluated. Although this line of research is very promising and would greatly benefit practical deployment, the challenge of making sure that the user model truly reflects what real users are likely to do, which oftentimes is dependent on very subtle aspects of the dialogue design and task domain, is a daunting task.

As noted in section 3.1, having an explicit objective function can be advantageous in that it can serve as an evaluation metric. Oftentimes, that metric is difficult to realize without the aid of user simulation. User simulation provides a testing environment for conducting controlled experiments which might be too burdensome for real users. Unfortunately, just because a SDS does well with respect to average or total reward in simulations does not guarantee that real users will "reward" the system accordingly. In fact, although the objective function is supposed to globally optimize the dialogue, it has never really been empirically evaluated against systems that optimize local (myopic) decisions. Local optimization may provide a better user experience in cases where users unexpectedly change their approach to responding to the system; that is, when local rewards change. For example, when users become suddenly frustrated, a SDS that is focused on local decisions may be better prepared to take actions that mollify and keep them engaged. Of course, $R$ can be a function of user frustration as well, but, as we discussed in section 3.2, that may or may not be feasible.

The question is, how well do the simple reward functions that are commonly used within the reinforcement learning framework reflect real users' reaction to a SDS? After all, depending on the objective function, which itself can be suspect, the cost of some types of errors, such as misunderstandings, can be worse than others, such as false rejections [22]. The best practice of course is to conduct user studies in addition to simulation experiments, which, because of lack of time and resources, is not often pursued by researchers, with notable exceptions [4].

In general, current practices for evaluating reinforcement learning-based systems need to be scrutinized more carefully. A big concern is the common practice of testing policies that have been trained on a simulated user using the *same* simulated user. This is essentially cheating. As pointed out in [23], policies trained with a poor user simulation model may appear to perform well when tested on the same model, but fail when tested on a better user simulation model. Fortunately, the converse was not true: policies learned with a good model will still perform well when tested on a poor model. Another common practice is to evaluate reinforcement learning policies against hand-crafted solutions (e.g., from an existing SDS) using average or total reward as a metric. The problem is that the hand-crafted solutions are not typically optimized according to same objective function, so it is not a fair comparison. If, for example, a learned policy is evaluated against a hand-crafted confidence threshold, then that threshold should be tuned to maximize the expected cumulative reward.

## 4. Discussion

In this section, we attempt to weigh the strengths described in section 3 against the weaknesses, and arrive at a summary judgment. We note that the opinions expressed here are exactly that, opinions, and nothing more.

Perhaps the strongest argument for the reinforcement learning approach to dialogue management is that it is statistically driven and theoretically principled. The approach models the uncertainties of spoken dialogue (which compared to non-statistical approaches, offers benefits in-and-of-itself), makes explicit exactly what is being optimized, and applies mathematically rigorous techniques to derive an optimal policy. As the approach evolves and dialogue optimization is tightly coupled with speech recognition and spoken language understanding, it is likely that performance will greatly increase and a stronger push will be made to make the various "manual" aspects, such as state space selection, more standardized. Even with the current state-of-the-art, given that application developers accept the "default" settings, what they have to gain — namely, a first stab at a policy, is tremendous. Handcrafting rules or heuristics for dialogue management is very difficult and time consuming. Even if application developers choose not to use the generated policy, it is certain to give them insight into how they should craft their own strategies. Of all the approaches to dialogue management out there, we believe that the reinforcement learning approach offers the best hope of automating dialogue management.

On the other hand, perhaps the strongest argument against the reinforcement learning approach with respect to practical deployment has little to do with the approach and more to do with the business of building a commercial SDS. Even with many of the challenges described in the previous sections solved, it is unlikely that application developers will quickly move to adopt the approach. After all, they have customers to please, and modifying policies while still remaining within the reinforcement learning framework is currently overly complex and inaccessible to most application developers. The only way application developers will give up their control over a fundamental aspect of their business is to be empirically convinced that the approach *always* creates policies that outperform their human-engineered policies for every domain. This is a tall order and requires winning over the trust of not only the developers, but their customers as well. That way, if a customer ever wants to modify a SDS so that it behaves differently than what would be prescribed by the optimal policy, the developer could always show them empirically that the preferred way results in better performance, and ultimately, more revenue for the customer.

So, is reinforcement learning currently ready for practical deployment? Probably not. Could it ever be? Yes, but it seems to be a long road ahead. That said, there remains one more criticism that can be leveled against the reinforcement learning approach. Unlike many dialogue management models based upon discourse, the reinforcement learning approach offer no theoretical insight into how various processes and phenomena of discourse, such as pronoun resolution, operate. All that the approach does is learn an optimal controller for whatever data it receives. Linguistic theory is mostly absent, or stuffed into the selection of the state space or reward function, which may be disadvantageous in the long run as the types of dialogues that application developers want to build systems for involve greater and greater task complexity and discourse structure. Because similar criticisms had been leveled against statistical approaches to speech recognition decades ear-

lier, perhaps the only response possible at this point is that only time, and verifiable results, will tell.

## 5. Conclusion

In this paper, we investigated reinforcement learning methods that utilize a fully or partially observable MDP for dialogue management. We assessed the strengths and weaknesses of these methods with respect to practical deployment and discussed challenges that need to be overcome before these methods can become commonplace in deployed systems. Finally, we compared the primary strength of these methods against its primary weakness and concluded that the current state-of-the-art is not quite ready for practical deployment.

## 6. Acknowledgments

This position paper resulted from illuminating discussions with Microsoft colleagues, including Max Chickering, Li Deng, Stephen Potter, and Ye-Yi Wang. Kristiina Jokinen gave valuable feedback. Thanks especially to Dan Bohus who provided very helpful insights and suggestions.

## 7. References

[1] R.S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[2] C.J.C.H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[3] E. Levin, R. Pieraccini, and W. Eckert, "Using markov decision processes for learning dialogue strategies," in *In Proceedings of the IEEE Transactions on Speech and Audio Processing*, 1998, vol. 8, pp. 11–23.

[4] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing dialogue maangement with reinforcement learning: Experiments with the NJ-fun system," *Journal of Artificial Intelligence Research*, vol. 16, pp. 105–133, 2002.

[5] S. Young, "The statistical approach to the design of spoken dialogue systems," 2002.

[6] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning.," in *ACL*, 2000.

[7] J.D. Williams, P.Poupart, and S. Young, "Factored partially observable markov decision processes for dialogue management.," in *4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, August 2005.

[8] B. Zhang, Q. Cai, J. Mao, and B. Guo, "Planning and acting under uncertainty: A new model for spoken dialogue systems," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2001, pp. 572–579.

[9] L.P. Kaelbling, M.L. Littman, and A.P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[10] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[11] E.A. Hansen, "An improved policy iteration algorithm for partially observable MDPs," in *Advances in Neural Information Processing Systems*, Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, Eds. 1998, vol. 10, The MIT Press.

[12] M.A. Walker, R.J. Passonneau, and J.E. Boland, "Quantitative and qualitative evaluation of DARPA communicator spoken dialogue systems," in *Proceedings of the ACL-2001*, 2001, pp. 515–522.

[13] T. Paek, "Empirical methods for evaluating dialog systems," in *Proceedings of the Workshop on Evaluation for Language and Dialogue Systems*, 2001, pp. 1–8.

[14] D. Litman and S. Pan, "Designing and evaluating an adaptive spoken dialogue system," *User Modeling and User-Adapted Interaction*, vol. 12, no. 2/3, pp. 111–137, 2002.

[15] A.Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. 17th International Conf. on Machine Learning*, 2000, pp. 663–670.

[16] T. Paek and D.M. Chickering, "On the markov assumption in spoken dialogue management," in *6th SIGDIAL Workshop on Discourse and Dialogue*, 2005, pp. 35–44.

[17] D.M. Chickering and T. Paek, "Personalizing influence diagrams: Applying online learning strategies to dialogue," *User Modeling and User-Adapted Interaction*, vol. To appear, 2006.

[18] J. D. Williams and S. Young, "Scaling up pomdps for dialog management: The "summary pomdp" method.," in *IEEE Workshop Automatic Speech Recognition and Understanding (ASRU)*, 2005.

[19] A.G. Barto, S.J. Bradtke, and S.P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1-2, pp. 81–138, 1995.

[20] D. Bohus and A. Rudnicky, "Error handling in the ravenclaw dialog management system," in *Proceedings of HLT-EMNLP*, 2005.

[21] W. Eckert, E. Levin, and R. Pieraccini, "User modeling for spoken dialogue system evaluation," in *Proceedings of IEEE ASRU Workshop*, December 1997.

[22] D. Bohus and A. Rudnicky, "A principled approach for rejection threshold optimization in spoken dialog systems," in *Proceedings of Interspeech*, 2005.

[23] J. Schatzmann, M. Stuttle, K. Weilhammer, and S. Young, "Effects of the user model on simulation-based learning of dialogue strategies.," in *IEEE Workshop Automatic Speech Recognition and Understanding (ASRU)*, 2005.