Learning Mixed Initiative Dialog Strategies By Using Reinforcement Learning On Both Conversants

Michael S. English and **Peter A. Heeman**
Center for Spoken Language Understanding
OGI School of Science & Engineering
Oregon Health & Science University
Beaverton OR, 97006, USA
menglish6@gmail.com and heeman@cslu.ogi.edu

Abstract

This paper describes an application of reinforcement learning to determine a dialog policy for a complex collaborative task where policies for both the system and a proxy for a user of the system are learned simultaneously. With this approach a useful dialog policy is learned without the drawbacks of other approaches that require significant human interaction. The specific task that the agents were trained on was chosen for its complexity and requirement that both conversants bring task knowledge to the interaction, thus ensuring its collaborative nature. The results of our experiment show that you can use reinforcement learning to create an effective dialog policy, which employs a mixed initiative strategy, without the drawbacks of large amounts of data or significant human input.

1 Introduction

The problem of developing a dialog manager can be expressed as the task of building a specific dialog policy for the dialog system to follow as it interacts with the user. A dialog policy can be thought of as an enumeration of all of the states a dialog system can be in, and the corresponding action to take from each of those states. Thus a policy completely specifies the behavior of a dialog manager.

Most conventional approaches to accomplishing this task seek to directly model human interactions in some manner. These techniques include handcrafting a policy, using a Wizard-of-Oz approach in an iterative manner and inducing a policy from a human-human dialog corpus. All three approaches have shortcomings that make them less than ideal for developing dialog systems. The approach of handcrafting of a dialog policy is problematic as it is difficult to predict how a user with interact with it, making it difficult to craft an optimal policy. To get around this, an iterative approach can be used, with a Wizard taking the place of the system. However, it is still difficult to train a wizard, and it is difficult to explore many different strategies in order to find the optimal one. Human-human dialog can be used for policy generation, as this should represent optimal behavior to accomplish a task. However, computers are not capable of behaving exactly as a human. In addition, humans might not interact with a computer as they would another person.

Recently a number of researchers have proposed using reinforcement learning to alleviating the problems encountered with more conventional methods of developing dialog policies. With the development of a good policy evaluation function, reinforcement learning can effectively and quickly explore a large policy space. There is the additional benefit that it will learn a policy that is optimal for the capabilities of the system.

The main drawback of reinforcement learning approaches is that they require some form of conversational partner to train the system against. Conventionally, these partners have taken the form of a human (Walker, 2000; Singh et al., 2002) or a simulated user (Levin et al., 2000; Scheffler and Young, 2002; Georgila et al., 2005). These two types of conversational partners limit the complexity and diversity of policies that can be generated by reinforcement learning. These two approaches to training partners limit the whole system to the abilities of the partners themselves. For a human partner we

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 1011–1018, Vancouver, October 2005.

run into the significant time and effort problems that were present in Wizard-of-Oz and handcrafting policy development. With a simulated user the system is limited by the complexity and flexibility of the simulated user, which itself can require a large degree of handcrafting by its creator.

In this paper, we propose a solution to the conversational partner problem of generating a dialog policy with reinforcement learning. We have taken a complex collaborative task and used reinforcement learning, applied to both participants, to develop a dialog policy for the task. By training both agents simultaneously we are able to avoid the uncertainties of creating a user to train against, as well as the time and data limitations of training directly against humans. Our training approach allows us to avoid these conventional drawbacks even while applying reinforcement learning to complex tasks.

Section 2 provides a brief overview of previous work in using reinforcement learning for dialog systems. Sections 3 and 4 describe the dialog task and its specification as a reinforcement-learning problem. Section 5 and 6 present the results of this experiment and a discussion of them.

2 Related Work

A number of researchers have explored using reinforcement learning to create a policy for a dialog system. Walker (2000) trained a dialog system, ELVIS, to learn a dialog strategy for supporting spoken language access to a user's email. The main function of ELVIS is to provide verbal summaries of email folders. This summary could consist of simple statements about the number of messages or a more detailed description of current emails.

Reinforcement learning is used to determine the best settings for a variety of properties of the system. For example, the system must learn to choose between email reading styles of reading back the full email first, reading a summary of the email first, or prompting the user with the two choices of reading styles. The system also learns whether it is better to take a mixed initiative or a system initiative strategy when interacting with the user.

To enable the learning process, ELVIS utilized human users as its conversational partner. Users performed a set of tasks with ELVIS, with each run using different state-property values, which were randomly chosen for that dialog. In order to support humans as a training partner Walker restricted the policy space so that it would only contain policies that were capable of accomplishing the available system tasks. Thus, during training the users would not be faced with a system that simply could not perform the tasks asked of it.

ELVIS was trained with a Q-learning approach that sought to determine the expected utility at each state, where utility was a subjective function involving such variables as task completion and user satisfaction. The state variables utilized in the training process were (a) whether the user's name is known, (b) what the initiative style is, (c) the task progress, and (d) what the user's current goal is. Given these state variables, ELVIS was able to learn the best style to adopt in responding to the user's requests at various points in the dialog. One major shortcoming of the conversational partner used with ELVIS is its reliance upon human interaction for training. This shortcoming is somewhat mitigated by the fact that the learning problem was one of fitting together preexisting policy components, but would be severely limiting if the goal was to learn a complete dialog policy. The amount of data necessary for learning a complete policy makes direct human interaction in the learning process unrealistic.

Levin et al. (2000) tackles a slightly different reinforcement-learning task. She is learning a policy to use in a dialog system built from a small set of atomic actions. This system is trained to provide a verbal interface to an airline flight database. This system is able to provide users with a way to find flights that meet a dynamic set of criteria. The dialog agent's state consists of information regarding the departure city, destination city, flight date, etc. Levin takes a useful approach in reducing the size of true state space by simply tracking when a particular state variable has a value rather than including the specific value in the state. For instance during a dialog when the system determines that the departure city is New York it does not distinguish this from when it has determined that the departure city is Chicago.

To converse with the dialog agent during reinforcement learning, Levin uses a "simulated user." The simulated user is created from a corpus of human dialogs with a prior airline system. In de-

veloping this user Levin makes the simplifying assumption that a user's response is based solely on the previous prompt. Then the specific probabilities for each user response are determined by examining the corpus for exchanges that match the possible prompts for the new dialog agent as well as hand crafting some of the probabilities. During the actual learning the agent used Monte Carlo training with exploring starts in order to fully explore the state space.

The "simulated user" method of supplying the conversational partner seems difficult and not particularly applicable to tasks where a dialog corpus does not already exist, but Kearns and Singh (1998) indicates that the accuracy of the transition probabilities for the probabilistic user is not critical for the dialog agent to learn an optimal strategy. While this experiment does allow for the dialog agent to learn a complex strategy, the notion of learning against a simulated user limits the space of policies that will be considered during training. Training against a conversational partner that is a model of a human automatically prejudices the system towards policies that we would be inclined towards building by hand and precludes the sincere exploration of all possible policies.

3 Task Specification

For our experiment we use the task presented in Yang and Heeman (2004), which is a modification of the DesignWorld task of Walker (1995). The task requires 2 conversants to agree on 5 pieces of furniture to place in a room. Both conversants know all of the furniture items that can be chosen, which differ by color, type and point value. Each conversant also has private preferences about which furniture items it wants in the room; such as 'if there is a red couch in the room, I also want a lamp in the room'. Each preference has a score. As this is a collaborative task, the conversants have the goal of finding the 5 furniture items that have the highest score, where the score is the sum of the point value of each of the 5 chosen furniture items less the scores for any violated preferences of either conversant.

The conversational agents work to achieve their goal by performing the following actions: **propose**, **accept**, **reject**, **inform**, and **release turn**. If there is not a current proposal, either agent can **propose** an item, which makes that item into the current proposal. If there is a current proposal, the other conversant can **accept** it or **reject** it. Accepting an item results in that item being included in the task solution and removes it as the current proposal. Rejecting a proposed item removes it as the current proposal. When an item has been rejected it remains a valid choice for future proposals. In addition to accepting or rejecting a proposal, either conversant may **inform** the other conversant of preferences that are violated by the current proposal. A preference is violated by the current proposal if the addition of that proposed item to the solution set would cause the solution set to violate the preference. When a conversant informs of a violated preference, that preference becomes mutually known and so affects future decisions by both participants. Only preferences that are not known by the other conversant are communicated. For turn taking, we include the action **release turn**, which the conversant that currently has the turn can perform to signal that it is relinquishing the turn (cf. Traum and Hinkelman, 1992). Note that after a release turn, the other agent must make the next move, which could itself be a release turn. The inclusion of this action allows conversants to perform multiple actions in a row, such as a reject, an inform, and a propose. Our approach to turn taking differs slightly from Yang and Heeman, as they make it an implicit part of other actions.

In order to successfully utilize these actions in a dialog, some reasoning effort is required of the conversants. Conversants must be able to determine what preferences are violated by a pending proposal and which of the remaining items makes the best proposal. In order to keep the reasoning effort manageable, we follow Yang and Heeman and use a greedy algorithm to pick the item that results in the best score for the item plus the set of items already accepted. The conversants do not consider interactions with the items that will be subsequently added to the plan. Conversants using this greedy approach can construct a plan that is very close to optimal.

4 Learning Specification

4.1 Agent Specification

In order to apply reinforcement learning to this task we must formalize the conversants as reinforcement

learning agents, specifying their state and actions, as well as the environment they will interact in. In order to reduce the size of the state space for this task we simplified the representation of the state in a manner similar to that done by Levin (2004). We formulated the state of the dialog agents with many of the more specific details of the actual state of the task removed. For instance the agent state does not include specific information about the furniture item that is the pending proposal, rather the agent's state only indicates that there is a pending proposal.

The state specification for each agent includes the following binary variables: **Pending-Proposal**, **I-Proposed**, **Violated-Preference**, **Prior-Violated-Preferences**, and **Better-Alternative**. **Pending-Proposal** indicates whether an item has been proposed but not accepted or rejected. **I-Proposed** indicates if the agent made the most recent proposal. **Violated-Preference** indicates that the pending proposal has caused one or more violations of the conversant's private preferences. **Prior-Violated-Preferences** indicates whether the conversant had one or more violated preferences when the pending proposal was made. This variable allows the agent to remember what its original response to a proposal was, even after it may have shared all of its preferences that were violated (thus creating a state where it no longer has any violated personal preferences). **Better-Alternative** indicates that the agent thinks it knows an item that would achieve a better score than the item currently proposed.

The actions from Section 3 can be sequenced in a number of different orders, leading to different policies. Unlike Yang and Heeman, who compared handcrafted policies, we use reinforcement learning to learn policy pairs, one part of the pair for the system, and the other for the simulated user. We have restricted the space of policies that can be learned. First, we reduce the space by only considering legal sequences of actions. For example, if there is a pending proposal, another item cannot be proposed. Second, after 5 items have been accepted, the dialog is automatically ended. Third, to keep the space of dialog policies small, we force an inform to inform of all violated preferences at once.

The Reinforcement Learning states and actions of our dialog agents capture a subset of the true state of the dialog. Our agents do not have the ability to distinguish between, or develop distinct policies in response to, the proposal of a blue chair versus a red desk. Since our formulation of the dialog agents do not encode specific information about items or preferences, the dialog environment must maintain these details. This extra information that must include the currently proposed item, what each agent's private and currently violated preferences are, what preferences are shared between each agent, what items have been accepted as part of the task solution, and what items are still available for selection. This technique of generalizing the state space is the same as the one used by Levin (2000), and allows us to keep the state space at a manageable size for our task.

4.2 Reinforcement Learning

For our Reinforcement Learning algorithm we chose to use an on-policy Monte Carlo method (Sutton and Barto, 1998). Our chosen task is naturally episodic since the two agents agreeing upon five items indicates task completion and thus the end of the dialog, which constitutes one learning episode. We also imposed a limit of 500 interactions per dialog in order to ensure that each learning episode was finite even if the task was not successfully completed. For some state-action pairs our task does not allow the accurate specification of the resulting state. In fact, due to the way that our state representation simplifies the true task environment an action choice for many states will necessarily lead to different states depending upon the task environment. For instance, proposing an item will sometimes lead to that items acceptance and sometimes it will be rejected. Given this uncertainty our learning approach necessarily had to learn the expected rewards of actions instead of states.

At the end of each dialog the interaction is given a score based on the evaluation function and that score is used to update the dialog policy of both agents. The state-action history for each agent is iterated over separately and the score from the recent dialog is averaged in with the expected return from the existing policy. We chose not to include any discounting factor to the dialog score as we progressed back through the dialog history. The decision to equally weight each state-action pair in the dialog history was made because an action's contribution to the dialog score is not dependent upon its

proximity to the end of the task. An action that accepts a proposed item at the beginning of the dialog should be rewarded as much as an action that accepts a proposed item later in the same dialog.

In order for the learning agents to obtain a large enough variety of experiences to fully explore the state space some exploration technique must be used. We chose to use e-greedy action selection in order to achieve this goal. With this approach the dialog agent makes an on policy action choice with probability 1-e and a random valid action choice the rest of the time.

Training both agents simultaneously causes each agent to learn its policy as an optimal response to the opposing agent. This can create problems in the initial stages of training as each agent has an immature policy that is based on little experience. In this situation each of the agents will associate weights with state action pairs based on action choices of the opposing agent that are themselves not well developed. As training progresses the eccentricities of the initial immature policies are perpetuated and the learning process does not converge on an effective dialog policy for either agent.

In order to combat the problem of converging to an effective policy we divided up the agent training process into multiple epochs. Each epoch is composed of a number of training episodes. The initial epsilon value is set to a large value and for each successive epoch the epsilon value for action selection is decreased. With an initially high epsilon value the agents are able to develop a policy that is initially weighted more heavily towards a response to random action selection than the immature policy of the other agent. As the epsilon value decreases, each agent slowly adjusts its learning to be weighted more heavily towards a response to the other agent's policy. This approach allows the agents to develop a minimally coherent dialog policy before beginning to rely too heavily upon the response of the opposing agent.

Utilizing this strategy of continuously decreasing epsilon values we were able to get both agents to converge to an effective and coherent dialog policy. The initial epsilon value was set to 80

4.3 Objective Function

In the reinforcement learning process the objective function provides the dialog agents with feed-back on the success of each dialog. The specification of this function requires input from a human. For our learning specification we crafted a simple function that attempted to model a human perception of a dialog's quality. Our objective function is linear combination of the solution quality (S) and the dialog length (L), taking the form:

$$o(S, I) = w_1 S - w_2 L$$

where w_1 and w_2 are positive constants. As higher values for S and lower values for L indicate better dialogs, we subtract $w_2 L$ from $w_1 S$. Instead of attempting to hand pick the constants in the objective function, we explored the effects of different values, which we report in Section 5.2.

For our experiment we trained the dialog agents for 200 epochs, where each epoch consisted of 200 training episodes. After the training the agents, we then had them perform 5000 dialogs with 100% on-policy action selection (i.e. strictly following the learned policy). The results of these 5000 dialogs were then combined to obtain an average plan score and average number of interactions for the policy of the agents. These two values are then combined according to the objective function to obtain a numeric score for the learned policy.

5 Results

In this section, we present the results of the dialog policies that we learned. We first present 3 baseline policies to which we will compare the performance of our learned policies. We will then present results varying the weights in the objective function in comparison to the baseline policies. As we are learning a pair of policies—one for the system and one representing the user—we explore how well the system policy does against handcrafted ones, that will represent what a user might do, rather than test it against its learned counter-part.

5.1 Baseline Policies

In order to provide comparative data to evaluate the effectiveness of our approach, we will compare the performance of the policies learned for the system and user against several pairs of handcrafted poli-

cies. The first pair implement the **unrestricted** initiative strategy of Yang and Heeman. Here, one conversant, A, proposes an item and then the other, B, informs A of any violated preferences. B then proposes an alternative and A informs B of any violated preferences. The process repeats until an item is proposed that does not violate any of the other agent's preferences. The second pair of policies implement the **restricted** initiative policy of Yang and Heeman, in which A proposes an item and B informs A of any violated preferences. However, the conversants do not switch roles: it is always A who proposes items and B that informs of preferences and accepts. These two policies represent successful handcrafted pairs of dialog policies. The third pair represents a minimum performance: A proposes an item and B simply accepts it. This is repeated for all 5 items, with A making all of the proposals. This policy is an **un-collaborative** approach, which represents how well A can do on its own.

5.2 Impact of Weights on Learned Policy

We first explore the ability of the reinforcement learning algorithm to learn a dialog policy pair that is optimal with respect to the objective function. The only important aspect of the weights is the ratio between the two: w_2/w_1. We varied the ratio from 0.1 to 0.5 in increments of 0.02. For each weight setting, we learned 66 policy pairs, and tested each policy pair on 1000 different task configurations. We compared the average objective function score of the learned policy pairs with the baseline restricted policy pair (cf. Scheffler and Young, 2002). Figure 1 shows the percentage of the learned policies that perform at least as well as the unrestricted policy pair

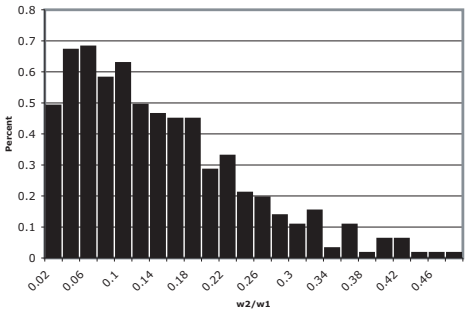

Figure 1: Percentage of learned policies performing better than unrestricted baseline pair.

at each weight setting. Interestingly, it is clear that there is a lack of convergence in the learning process, no weight ratio learns a good policy 100% of the time. Additionally, we see that as the weight ratio increases (putting more emphasis on shorter dialogs), the ability of the algorithm to learn good policies decreases. As the objective function gives this aspect more weight, it is more difficult for the objective function to learn the importance of solution quality. We think this lack of convergence is due to learning both the system and a simulated user at the same time, which is a more difficult reinforcement learning problem than just learning the policy for the system against a fixed user.

5.3 Lack of Convergence

To better understand the lack of convergence, we explore when a single weight is chosen for the objective function. For this analysis, we restricted ourselves to the objective function having a ratio for w_2/w_1 of 0.1, one of the best performing weights from section 5.2. For this setting, we learned a number of policy pairs, each learned from a different sequence of task configurations. We then tested each policy pair on 1000 task configurations, in which actions are selected strictly according to the learned policy. This gives us 1000 dialogs for each policy pair. We then computed the average objective function score for each policy pair and plotted them as a histogram in Figure 2. As can be seen, at this weight setting, 63% of the learned policies achieved an objective function score around 44.8. However, the rest achieved a performance substantially less than this. Hence, the reinforcement learning procedure does not always converge on an optimal solution.

To better understand why reinforcement learning is not always converging, we examined the components of the objective function score: solution quality and dialog length. Figure 3 uses the same x-axis as Figure 2: average objective function score. The y-axis plots the average solution quality and average dialog length. We see that at this weight ratio, all learned dialogue pairs are very consistent in solution quality, but that the difference in objective function scores is mainly due to differences in dialog length. This is consistent with our earlier observation that the reinforcement learning strategy sometimes disproportionately favors shorter dialog length.

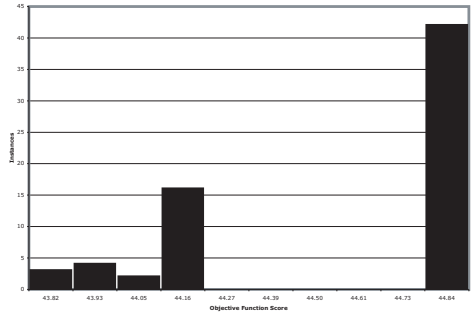

Figure 2: Average objective function scores for policies learned with w_2/w_1 = 0.1.

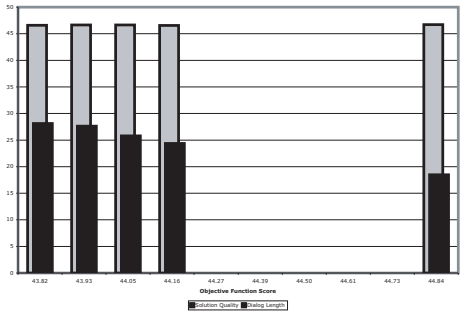

Figure 3: Variation of solution quality and dialogue length versus objective function score for policies learned with w_2/w_1 = 0.1.

5.4 Consistency of Policies

For the weight ratio of 0.1, the reinforcement learning algorithm usually finds a good policy pair. To further improve the likelihood of this happening, we could learn multiple policy pairs, and then pick the best performing one. In this section, we compare learned policies chosen in this way against the restricted baseline pairs. We learned 10 sets of 10 dialogue pairs. We then ran each on 1000 task configurations and chose the best performing policy pair in each set. We then ran the resulting 10 policy pairs on another set of 1000 task configurations. Table 1 gives the average objective function score for each of the 10 learned policy pairs and the 3 baseline pairs. From the table, we see that the learned policy pair performs almost as well as the restricted policy pair, for both solution quality and dialog length.

5.5 Robustness of Learned Policies

All of the results so far have used the learned policy for the system interacting with the corresponding policy that was learned for the user. However, there is no guarantee that a real user will behave like the learned policy. Thus, the true test of our approach is to run the learned system policy against actual users. The problem with testing our policies against actual users is that there are a number of aspects of dialog that we have not modeled, such as non-understandings, misunderstandings, and even parsing sentences into the action specification and generating sentences from the action specification. Thus, as a simplification we tested our learned system policy on the handcrafted baseline policies.

	Objective Function	Solution Quality	Dialog Length
Learned Policies	44.90	46.71	18.17
Restricted	45.04	46.89	18.44
Unrestricted	44.40	46.80	24.07
Uncollaborative	32.52	33.62	11.00

Table 1: Comparison of Learned Policies

For the weight ratio of 0.1, we learned 10 sets of 10 pairs of policies and choose the best policy pair from each set. For each of the 10 policy pairs, we ran the system policy against the 6 individual policies from the 3 baseline policy pairs. We changed the hand-crafted policies slightly from Yang and Heeman so that the policies would not fail if they encountered unexpected input. For example, for the restricted policy for A (the conversant who proposes but never informs), if the learned policy proposes an item, A always rejects it. For the restricted policy for B (the conversant who informs but never proposes), if the learned policy releases the turn when there is not an item proposed, B simply releases the turn back to the learned policy.

Figure 4 shows the resulting average objective function scores on 1000 dialog runs. For each baseline policy, we show the performance with the policy pair, and then with each side of the baseline policy interacting with the learned policy. We see that although the performance of the learned policy is not as good as with the handcrafted pair, the performance is close, with the major shortcoming being a general increase in dialog length. Thus, the policies that we have learned our robust against different strategies a user might want to use.

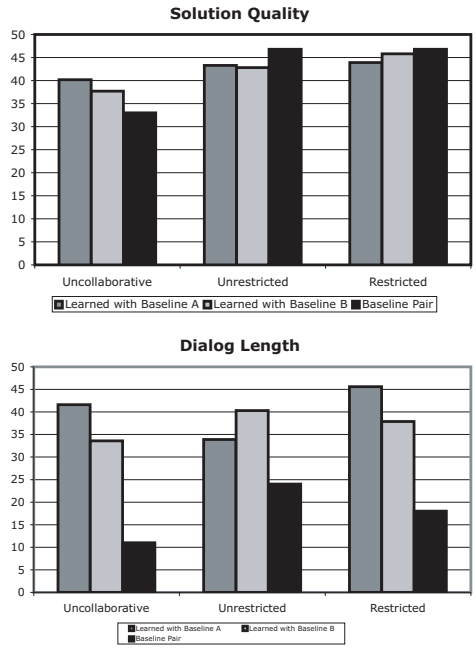

Figure 4: Learned dialogue policies interacting with baseline policies.

6 Conclusion

In this paper, we proposed using reinforcement for learning a dialog strategy for the system. Our approach differs from past research in that we learn the system policy in conjunction with learning a user policy. This approach of learning the user policy allows us to minimize human involvement, as neither a training corpus must be collected nor a simulated user built. Thus, the only human input required for this approach was to define the domain task and to define success in that domain. While our training approach did not always find an effective policy, we overcame this obstacle by carefully choosing a ratio for the weights in the objective function and by running the learning algorithm multiple times. Our approach resulted in learned system and user dialog policies that achieved comparable performance with handcrafted system and user policy pairs. Furthermore, the learned system policies were robust. When the learned system policies 'conversed' with the handcrafted user policies, the resulting dialogs had comparable solution quality to what the handcrafted system and user policies achieved together.

Even with the lack of convergence our approach could be applied to more complicated domains in order to learn an effective dialog policy. Our approach would be especially useful in situations where there are no existing corpora of human-human interactions for the domain or as a way to provide a check against a policy based on human intuition. In most situations where the domain requires significant collaboration between the dialog system and the user, training both the system and a user simultaneously will prove to be much less costly and labor intensive approach.

7 Acknowledgments

The authors thank John Moody and Fan Yang for helpful discussions. Partial funding for this research was provided by the National Science Foundation under grant IIS-0326496. The first author is now at Google.

References

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

M. Kearns and S. Singh. 1998. Finite-sample convergence rates for q-learning and indirect algorithms. In *NIPS*, Denver CO.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

K. Scheffler and S. J. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *HLT*, pages 12–18.

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. Optimizing dialogue managment with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. MIT Press, Cambridge MA.

D. Traum and E. Hinkelman. 1992. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599.

M. Walker. 1995. Testing collaborative strategies by computational simulation: Cognitive and task effects. *Knowledge-Based Systems*, 8:105–116.

M. Walker. 2000. An application of reinforcement learning to dialog strategy selection in a spoken dialogue system. *Journal of Artificial Intelligence Research*.

F. Yang and P. Heeman. 2004. Using computer simulation to compare two models of mixed-initiative. In *ICSLP*.