Empirical Evaluation of a Reinforcement Learning Spoken Dialogue System

Satinder Singh and **Michael Kearns** and **Diane J. Litman** and **Marilyn A. Walker**
AT&T Labs
180 Park Avenue
Florham Park, NJ 07932
{baveja,mkearns,diane,walker}@research.att.com

Abstract

We report on the design, construction and empirical evaluation of a large-scale spoken dialogue system that optimizes its performance via reinforcement learning on human user dialogue data.

Introduction

The formalisms of Markov decision processes (MDPs) and reinforcement learning (RL) have become a standard approach to many AI problems that involve an agent learning to improve performance by interaction with its environment (Sutton, 1991; Kaelbling et al., 1996). While the theory of these formalisms is quite advanced, applications have been limited almost exclusively to problems in control, operations research, or game-playing (e.g., Crites and Barto, 1995; Tesauro, 1995). In this paper, we describe an application of RL to a rather different type of problem, in which the MDP models a system's interaction with a population of human users, and RL is used to optimize the system's performance.

Figure 1: A block diagram representation of a spoken dialogue system.

We have adapted the methods of RL to the problem of automatically learning a good dialogue policy in a *spoken dialogue system* (SDS). The different components of an SDS are shown in block diagram form in Figure . In a typical SDS, the user speaks to the system in real time through the telephone, using free-form natural language, in order to retrieve desired information from a back-end database component. The user's speech is interpreted through an automatic speech recognition (ASR) component. The *dialogue policy* decides what the system should say (or in RL terminology, which *action* it should take), again in natural language, through a text-to-speech (TTS) component, at each point in the dialogue.

S1: Welcome to RLDS. How may I help you?
U1: What wineries in Lambertville are open in the morning? [ASR output: *what wineries in Lambertville open in the morning.*]
S2: Did you say you are interested in Lambertville?
U2: Yes.
S3: I found a winery near Lambertville that is open in the morning. It is ... Please give me feedback by saying 'good', 'so-so', or 'bad'.
U3: Good.

Figure 2: A transcription of an example spoken dialogue with our NJFun system. This dialogue happened to go very well, and is relatively short. In general, dialogue length varied between 3 and 20 exchanges between user and system.

Figure 2 shows the transcription of a sample spoken dialogue from the NJFun system we implemented to provide telephone access to a database of activities in New Jersey. In this dialogue, by starting with the open-ended greeting "How may I help you?", the system lets the *user* take the *initiative* in providing information about the activity they are interested in. User responses in such cases may be relatively unconstrained. In contrast, the *system* could take the initiative by saying the more restrictive phrase "Please tell me the location you are interested in", thus constraining the user to provide information about the location of the activity. Which of these contrasting choices of user or system initiative is superior may depend strongly on the properties of the underlying and imperfect ASR, the population of users, as well as the dialogue so far. This choice of initiative occurs repeatedly throughout a dialogue, and is but one example of a class of difficult design decisions.

The traditional, or what we shall call Monte Carlo, approach to learning dialogue policies from data is to pick a set of dialogue policies that experts intuitively feel are good, implement each policy as a separate SDS, collect data from many representative human users for each SDS, and then use standard statistical tests to pick the best dialogue policy and system according to some performance criterion or reward measure (the choice of which we will say more about later).

Only a handful of dialogue policies can be compared this way because of the cost and time of using human subjects. On the other hand, as we will show for our system, many thousands of dialogue policies may still be left after the experts have excluded all policies that are clearly suboptimal.

In this paper we show that the methods of RL are well-suited to the problem of searching the large space of policies that survive after pruning by experts. At a high level, our RL methodology involves the choice of appropriate reward measures and estimates for dialogue state, the deployment of an initial training system that generates deliberately exploratory dialogue data, the construction of an MDP model of user population reactions to different action choices, and the redeployment of the system using the optimal dialogue policy according to this model. This RL method is more data-efficient than the Monte Carlo method, because it evaluates actions as a function of state rather than evaluating entire policies.

While RL has been applied to dialogue system design in previous research (Biermann and Long, 1996; Levin et al., 1997; Walker et al., 1998; Singh et al., 1999), this paper provides a larger scale test of these ideas. We describe our design and implementation of the NJFun system and our controlled experiments with human users verifying our RL-based methodology. The results we describe here provide empirical evidence that, when properly applied, RL can quantitatively and substantially improve dialogue system policy. For example, one of our main results is that the rate of task completion rose from 52% in the training system to 63% in the learned system. In a companion paper (Litman et al., 2000), we describe the same basic system and experiment, but focus on details and analyses more relevant to computational linguistics (such as linguistic analyses of the learned policy, and novice versus expert performance).

Our system and experiments help focus attention on the many challenges spoken dialogue systems present to the prevailing theory and application of RL. These include the fact that the Markov property cannot be guaranteed in an application modeling human users, the difficulty of balancing the need for exploratory data with the need for a functioning training system, and the inherent difficulty of obtaining large amounts of training data in such applications.

Choices in Dialogue Policy

For our purposes, an ASR can be viewed as an imperfect, noisy sensor with an adjustable "parameter" (the *language model* or *grammar*) that can be tuned to influence the types of speech recognition mistakes made. In addition to any perceived matches in the utterance, the ASR also returns a score (typically related to log-likelihood under a hidden Markov model) giving a subjective estimate of confidence in the matches found. This score is important in interpreting the ASR results.

In this work, we concentrate on automating two important types of decisions faced in dialogue policy design, both of which are heavily colored by the ASR facts above. The first type, of which we have already seen an example, is choice of *initiative* — namely, whether the system at any given point should prompt the user in a relatively open-ended manner (often referred to as *user* initiative) or a relatively restrictive manner (*system* initiative).

The second type of choice we investigate is that of *confirmation*. After it has applied the ASR to a user utterance, and obtained a value for some attribute of interest (for instance, town = Lambertville), the system must decide whether to confirm the perceived utterance with the user. In Figure 2, for example, the system chooses to confirm the location but not the activity type (wineries) or the activity time (morning). While we might posit that confirmation is unnecessary for high values of the ASR confidence, and necessary for low values, the proper definitions of "high" and "low" would ideally be determined empirically for the current state (for instance, depending on whether there has been difficulty on previous exchanges), and might depend on our measure of system success.

In the NJFun system, we identified many different dialogue states for which we wanted to *learn* whether to take user or system initiative for the next prompt. Similarly, we identified many different dialogue states in which we wanted to learn whether to confirm the ASR-perceived user utterance, or not to confirm. The actual prompts used in each case were hand-coded; we learn only the choice of initiative and the choice of confirmation, not what natural language utterances to generate. We note that there is genuine and spirited debate over choices of initiative and confirmation among dialogue system designers (Danieli and Gerbino, 1995; Haller and McRoy, 1998, 1999; Smith, 1998; Walker et al., 1998), which is precisely why we wish to automate, in a principled way, the process of making such choices on the basis of empirical data.

RL for Dialogue Policy Design

In this section, we describe the abstract methodology we propose to apply RL to dialogue policy design. In the next section, we will describe in detail the instantiation of this methodology in the NJFun system.

In order to apply RL to the design of dialogue policy, it is necessary to define a *state-based* representation for dialogues. One obvious but impractical choice for this state is a transcript or system log of the entire dialogue so far, which would include the audio so far, the utterances matched by the ASR, the language models used, the confidence scores returned by the ASR, and perhaps many other quantities. In practice, we would like to compress this state as much as possible — representing states by the values of a small set of features — without losing information necessary for making good decisions. We view the design of an appropriate state space as *application-dependent*, and a task for a skilled system designer.

Given choices for the state features, the system designer can think in terms of the state space, and appropriate actions to take in each state. For some states, the proper action to take may be clear (for instance, greeting the user in the start state, or querying the database when all informational attributes are instantiated). For other states, the system designer may debate a choice of actions that may best be determined by *learning* (such as choices of initiative and confirmation). Each mapping from such choice-states to a particular action is a distinct *dialogue policy*.

We also assume that the system designer has chosen a particular *reward function* that can be measured with relative ease on any given dialogue, that takes on scalar values, and whose expectation over the user population is to be maximized. The subject of appropriate measures for dialogue system success is a complex one, and there are many natural choices (Danieli and Gerbino, 1995; Walker et al., 1998), including user satisfaction measures, measures of task completion, and sales figures (in commercial applications). In our empirical results, we commit to a task completion reward measure for the optimization, but also examine several other common reward measures.

Thus, our methodology requires as a starting point that the designer choose a state representation and a reward function, and for perhaps a large number of states, to identify a fixed number of actions to be chosen from. Suppose that the designer implements an initial dialogue policy, and collects a set of dialogues from a sample of the user population. Each dialogue, of course, is a sequence of alternating system and user utterances terminated by a scalar reward:

$$s_1 \rightarrow_{a_1,r_1} s_2 \rightarrow_{a_2,r_2} s_3 \rightarrow_{a_3,r_3} \cdots$$

where the notation $s_i \rightarrow_{a_i,r_i} s_{i+1}$ indicates that at the ith exchange, the system was in state s_i, executed action a_i, received reward r_i, and then the state changed to s_{i+1}. From many such sequences, we can estimate *transition probabilities* of the form $P(s'|s,a)$, which denotes the probability of a transition to state s', given that the system was in state s and took action a. Our estimate of this probability is simply the number of times, in all of the dialogues, that the system was in s, took a, *and* arrived in s', divided by the number of times the system was in s and took a (regardless of next state). Similarly, we can estimate a reward function that maps states and actions to rewards. For the reward functions we will examine, the rewards will be nonzero only at terminal states.

The estimated reward function and transition probabilities constitute a *Markov decision process* (MDP) model of the user population's interaction with the system[1]. It (hopefully) captures the stochastic behavior of the users when interacting with the system. Note that in order to have any confidence in this model, the training data must have tried many possible actions from many possible states, and preferably many times. In other words, the training data must be *exploratory* with respect to the chosen states and actions. Perhaps the most straightforward way of ensuring exploratory training data is to take actions randomly. While this is the approach we take, it requires that we be exceptionally careful in designing the actions allowed at each state, in order to guarantee that the random choices made always result in a dialogue sensible to human users. (Keep in mind that there is no exploration in states where the appropriate action is already known and fixed by the system designer.) Other approaches to generating exploratory data are possible.

The final step is to determine the optimal policy in the estimated MDP using a dynamic programming algorithm such as value iteration (e.g., Kaelbling et al., 1996), and then to implement this policy as the learned dialogue policy. To the extent that the estimated MDP is an accurate model of the user population, this final system should maximize the reward obtained from *future* users. Here is a summary of the proposed methodology:

- I. Choose an appropriate reward measure for dialogues, and an appropriate representation for dialogue states.
- II. Build an initial state-based *training* system that creates an *exploratory* data set. Despite being exploratory, this system should provide the desired basic functionality.
- III. Use these training dialogues to build an empirical MDP model on the state space.
- IV. Compute the optimal dialogue policy according to this MDP.
- V. Reimplement the system using the learned dialogue policy.

The NJFun System

In this section, we describe the functionality and construction of the spoken dialogue system on which we tested our methodology. The back-end database for our system contained information on interesting places to visit in New Jersey. The database was indexed by three keys: the activity type (such as historic sites, wineries, museums, etc.); the name of the New Jersey town in which the activity is located (such as Morristown or Lambertville); and the hours in which the place is open. As an example, the Liberty Science Center is indexed under activity type museum, location Jersey City, and hours 10 AM to 6 PM. We binned activities into 9 activity types, and there were 149 distinct database entries[2]. The goal of NJFun is to help the user find all database entries matching a given binding of the desired activity type, location, and period of the day (morning, afternoon, or evening). For any given binding of these three attributes, there may be multiple database matches, which will all be returned to the user.

The system represents the current state of any dialogue by the values of six different state features, whose possible values and meanings are described in Figure . At a high level, these state variables tell the system which attribute it is currently working on, whether it has obtained a value for this attribute, what the confidence in that value is, how many attempts have been made to get a value for the attribute, what type of ASR grammar was most recently used, and an indication of whether there have been difficulties in earlier portions of the dialogue. We note that this state representation, in the interests of keeping the state space small, deliberately ignores potentially helpful information about the dialogue so

[1] Note that this MDP model is at best an approximation. With a small set of state features, there will be the problem of hidden state or partial observability, for which the richer POMDP model is often more appropriate (e.g., Kaelbling et al., 1996). We leave the use of POMDP models to future work.

[2] To support continuous use, the system's functionality could be extended in a number of ways such as a larger live database and support for followup questions by the users.

Feature	Values	Explanation
Attribute	1,2,3	Which attribute is being worked on
Confidence/ Confirmed	0,1,2, 3,4	0,1,2 for low, medium, and high ASR confidence 3,4 for explicitly confirmed, and disconfirmed
Value	0,1	Whether value has been obtained for current attribute
Tries	0,1,2	How many times current attribute has been asked
Grammar	0,1	Whether open or closed grammar was used
History	0,1	Whether there was trouble on any previous attribute

Figure 3: State features and values.

far. For example, there is no state feature explicitly tracking the average ASR score over all user utterances so far, nor do we keep information about previous attributes [3].

With the state space precisely defined, we can now provide some more detail on the *policy class* we considered[4]. This policy class is obtained by allowing a choice of system or user initiative whenever the system needs to ask or reask for an attribute, and by allowing a choice of confirming or simply moving on to the next attribute whenever the system has just obtained a value for an attribute. For example, in any state in which the *tries* feature has the value 0 and the *attribute* feature has value 1 (which means we are working on activity type, and we have yet to prompt the user for a value for this attribute), the system has a choice of uttering the user initiative prompt "How may I help you", or the system initiative prompt "Please tell me the activity type". In the case of a choice of system initiative, the system has the additional choice of calling the ASR on the user utterance using either a *closed* grammar intended just for that attribute, or an *open* grammar that may correctly recognize information offered on other attributes as well. The open grammar is always used with a user initiative prompt, because the choice of the closed grammar does not make sense in that case.

As another example, choices in confirmation policy are available at states for which the *value* feature is 1 immediately following a prompt to the user for the current attribute. In these states, if the *confidence/confirmed* feature is 0,1 or 2, we allow a choice of whether to confirm the attribute value obtained from the ASR, or to accept the current binding and move on to the next attribute.

We will call the set of all deterministic mappings from the states in which the system has a choice to a particular, fixed choice the *policy class* explored in our experiment. The total number of unique policies in this class was approximately 2^{42}. In keeping with the RL methodology described above, our goal is to compute and implement an approximately optimal policy in this very large class on the basis of RL applied to exploratory training dialogues.

Experimental Methodology

In this section, we describe in some detail the controlled user experiments we conducted. The next section presents the empirical results of these experiments.

Our experimental subjects were 75 fellow employees not involved with the project. The subjects were divided into a training population of 54 people and a test population of 21 people. Although we took the precaution of roughly balancing the male/female, native/non-native and experienced/inexperienced fractions in the training and test sets, subsequent analyses indicated that system performance did not depend significantly on any of these factors. Subjects were not told their training/test classification nor the purpose of our experiments.

As dictated by Step II of the RL methodology above, we first built a *training* version of the system, using the state space and action choices outlined in the preceding section, that used *random exploration*. By this we mean that in any state for which we had specified a choice of system actions, the training system chose randomly among the allowed actions with uniform probability. We again emphasize the fact that the allowed choices were designed in a way that ensured that any dialogue generated by this exploratory training system was intuitively sensible to a human user, and permitted the successful completion of any task the system was intended to perform. Nevertheless, it is important to note that over their multiple calls to the system (see below), training users may have effectively experienced multiple dialogue policies (as induced by the random exploration), while test users experienced a single, fixed, deterministic policy.

We designed a set of six specific tasks each participant was to complete using either the training system or the test system. Each task had an associated web page containing a brief text description of the desired information the participant should obtain, as well as a user survey common to all six tasks[5].

The training participants attempted to complete the six tasks using the exploratory training system. These 54 users generated a total of 311 dialogues[6]. These dialogues were then annotated with an objective *binary task completion* reward function. Since system logs could be matched with which of the six tasks the user was attempting, it was possible to directly compute from the system logs whether or not the user had completed the task. By "completed" we mean binding all three attributes (activity type, location, and time of day) to the exact values specified in the task description given on the associated web page. In this way, each training dialogue was automatically labeled by a $+1$ in the case of

[3]The system of course stores the actual values of previous attributes for the eventual database query, but as these do not influence future dialogue policy in any way, they are not stored as state features.

[4]Greater detail on the policy class can be found in a companion paper (Litman et al., 2000).

[5]Some of these survey questions formed the basis for the subjective reward measures examined in the next section.

[6]The total number of dialogues is less than $54 \times 6 = 324$ because a few users failed to attempt all 6 tasks.

a completed task, or -1 otherwise. We note that this definition of task completion guarantees that the user heard all and only the database entries matching the task specifications. Relaxations of this reward measure, as well as other reward measures, are discussed in the next section.

Finally, the 311 training dialogues, labeled by task completion, were used to build an MDP according the RL methodology, and the optimal policy according to this MDP was computed and implemented as the (now deterministic) dialogue policy in the test system.

The test users carried out the same six experimental tasks using the test system. The primary empirical test of the proposed methodology is, of course, the extent and statistical significance of the improvement in the allegedly optimized measure (task completion) from the training to test populations. The next section is devoted to the analysis of this test, as well as several related tests.

Results

Perhaps our most important results are summarized in the first two rows of Figure . In the first row, we summarize performance for the *binary completion* reward measure, discussed in the preceding section. The average value of this reward measure across the 311 dialogues generated using the randomized training system was 0.048 (recall the range is -1 to 1), while the average value of this same measure across the 124 dialogues using the learned test system was 0.274, an improvement that has a p-value of 0.059 in a standard two-sample t-test over subject means.

Reward Measure	**Train**	**Test**	Δ	**p-value**
Binary Completion	0.048	0.274	0.226	0.059
Weak Completion	1.72	2.18	0.46	0.029
Reuse	2.87	2.72	-0.15	0.55
Easy	3.38	3.39	0.01	0.98
NJFun understood	3.42	3.52	0.1	0.58
What to say	3.71	3.64	-0.07	0.71
Web feedback	0.18	0.11	-0.07	0.42

Figure 4: Train versus test performance for various reward measures. The first column presents the different reward measures considered (see text for detail); the second column is the average reward obtained in the training data; the third column is the average reward obtained in the test data; the fourth column shows the difference between the test average and the train average (a positive number is a “win”, while a negative number is a “loss”); the fifth column presents the statistical significance value obtained using the standard t-test.

We next examine the performance improvement for a closely related reward measure that we call *weak completion*. In weak completion, if *any* attribute is actually bound to an incorrect value (for instance, if the place was bound to Morristown instead of Lambertville when the latter was specified for the task), a reward of -1 is received. If no attribute is actually bound to an incorrect value, the reward is equal to the number of attributes correctly bound (recall that unbound variables are assigned as don’t-care). The motivation for this more refined measure is that reward -1 indicates that the information desired was not contained in the database entries presented to the user, while non-negative reward means that the information desired was present, but perhaps buried in a larger set of irrelevant items for smaller values of the reward.

In the second row of Figure , we show the improvement in weak completion from training to test[7]. The training dialogue average of weak completion was 1.72 (recall the range is -1 to 3), while the test dialogue average was 2.18. Thus we have a large improvement, this time significant at the 0.029 level. We note that the policy dictated by optimizing the training MDP for binary completion (which was implemented in the test system), and the policy dictated by optimizing the training MDP for weak completion (which was not implemented) were very similar, with only very minor differences in action choices.

Policy	**# Trajs.**	**Emp. Avg.**	**MDP Value**	**p-value**
Test	12	0.67	0.534	
SysNoconfirm	11	-0.08	0.085	0.06
SysConfirm	5	-0.6	0.006	0.01
UserNoconfirm	15	-0.2	0.064	0.01
UserConfirm	11	0.2727	0.32	0.30
Mixed	13	-0.077	0.063	0.06

Figure 5: Comparison to standard policies. Here we compare our test policy with several standard policies using the Monte Carlo method. The SysNoconfirm policy always uses system initiative and never confirms; the SysConfirm policy always uses system initiative and confirms; the UserNoconfirm policy always uses user initiative and never confirms; the UserConfirm policy always uses user initiative and confirms; the Mixed policy varies the initiative during the dialogue. For each policy, the second column shows the number of consistent trajectories in the training data, the third column shows the empirical average reward on these consistent trajectories, the fourth column shows the estimated value of the policy according to our learned MDP, and the fifth column shows the statistical significance (p-value) of the policy’s loss with respect to the test policy. For all but the UserConfirm policy, the test policy is better with a significance near or below the 0.05 level, and the difference with UserConfirm is not significant.

Although these results indicate an improvement in moving from the randomized training policy to the optimized policy, it is natural to ask how our optimized system compares to systems employing a dialogue policy picked by a human expert. Although implementing a number of hand-picked policies, gathering dialogues from them, and comparing to our learned system would be time-consuming and

[7]We emphasize that this is the *improvement* in *weak* completion in the system that was designed to optimize *binary* completion — that is, we only fielded a single test system, but examined performance changes for several different reward measures.

expensive (and in fact, is exactly the methodology we are attempting to replace), our training system provides a convenient and mathematically sound proxy. Since our training dialogues are generated making *random* choices, any dialogue in the training set that is *consistent* with a policy π in our policy class provides an unbiased Monte Carlo trial of π. (This is easily verified formally.) By consistent we mean that all the random choices in the dialogue agree with those dictated by π. We can average the rewards over the consistent training dialogues to obtain an unbiased estimate of the return of π.

Figure compares the performance of our learned test system, on the binary completion reward measure, to 5 fixed policies in our class that are common choices in the dialogue systems literature, or that were suggested to us by dialogue system designers. We see that in 4 cases, our learned policy outperforms these standard policies near or below the 0.05 level of significance, and in one case it is essentially tied. (Not surprisingly, the fixed UserConfirm policy that fared best in this comparison is most similar to the policy we learned.) Thus, in addition to optimizing over a large class of policy choices than is considerably more refined than is typical, the RL approach outperforms a number of natural standard policies.

We next discuss a number of other reward measures that we did not optimize the test system for, but for which we nevertheless examined system improvement or degradation. The two measures considered so far, binary and weak completion, are *objective* reward measures, in the sense that the reward is precisely defined as a function of the system log on a dialogue, and can be computed directly from this log. In contrast, we also examined a number of *subjective* measures that were provided by the human user following each dialogue. Each dialogue task was accompanied by a web survey (see Figure **??**), on which we asked the user whether they would use the system again (the *Reuse* reward measure, values 1 (worst) to 5 (best)), whether they found the system easy to use (the *Easy* reward measure, values 1 to 5), whether they thought the system understood what they had said (the *NJFun understood* reward measure, values 1 to 5), whether they knew what they could say at each point in the dialogue (the *What to say* reward measure, values 1 to 5), and finally, whether their experience on this dialogue was good, bad, or neutral (the *Web feedback* reward measure, values -1, 0, and 1 respectively).

Since we did not optimize for any of these subjective measures, we had no *a priori* expectations for improvement or degradation, and indeed Figure shows we did not find statistically significant changes in the mean in either direction for these measures. However, we observed a curious *move to the middle* effect in that a smaller fraction of users had extremely positive or extremely negative things to say about our test system than did about the training system. Although we have no firm explanation for this phenomenon, its consistency (it occurs to varying degree for all 5 subjective measures) is noteworthy.

Let us briefly summarize where we are. Our empirical results have demonstrated improvements in the optimized task completion measures of a complex spoken dialogue system,

Please repeat (or give) your feedback on this conversation. (*good, so-so, bad*)

1. Did you complete the task and get the information you needed? (*yes, no*)
2. In this conversation, it was easy to find the place that I wanted.
3. In this conversation, I knew what I could say at each point in the dialogue.
4. In this conversation, NJFun understood what I said.
5. Based on my current experience with using NJFun, I'd use NJFun regularly to find a place to go when I'm away from my computer.

Figure 6: User survey.

and no statistically significant changes in a number of non-optimized subjective measures, but an interesting move to the middle effect.

# of Trajs.	**# of Policies**	**Corr. Coeff.**	**p-value**	**Slope**	**Inter.**
> 0	1000	0.31	0.00	0.953	0.067
> 5	868	0.39	0.00	1.058	0.087
> 10	369	0.5	0.00	1.11	0.11

Figure 7: A test of MDP accuracy. We generated 1000 deterministic policies randomly. For each policy we computed a pair of numbers: its estimated value according to the MDP, and its value based on the trajectories consistent with it in the training data. The number of consistent trajectories varied with policy. The first row is for all 1000 policies, the second row for all policies that had at least 5 consistent trajectories, and the last row for all policies that had at least 10 consistent trajectories. The reliability of the empirical estimate of a policy increases with increasing number of consistent trajectories. The third column presents the correlation coefficient between the empirical and MDP values. The fourth column presents the statistical significance of the correlation coefficient. The main result is that the hypothesis that these two sets of values are uncorrelated can be soundly rejected. Finally, the last two columns present the slope and intercept resulting from the best linear fit between the two sets of values.

The skeptic might wonder if we have simply been fortunate — that is, whether our MDP might have actually been a rather poor predictor of the value of actions, but that we happened to have nevertheless chosen a good policy by chance. As some closing evidence against this view, we offer the results of a simple experiment in which we randomly generated many (deterministic) policies in our policy class. For each such policy π, we used the training dialogues consistent with π to compute an unbiased Monte Carlo estimate $\hat{R}_\pi$ of the expected (binary completion) return of π (exactly as was done for the hand-picked "expert" policies in Figure). This estimate was then paired with the value R_π of π (for the start state) in the learned MDP. If the MDP were a perfect model of the user population's responses to system actions, then the Monte Carlo estimate $\hat{R}_\pi$ would simply be a (noisy) estimate of R_π, the correlation between these two quantities would be significant (but of course dependent on the number of samples in the Monte Carlo estimate), and the best-fit linear relationship would be simply $\hat{R}_\pi = R_\pi + Z$ (slope 1 and intercept 0), where Z is a normally distributed noise variable with adjustable mean and variance decreasing as the number of consistent trajectories increases. At the other extreme, if our MDP had no relation to the user population's responses to system actions, then $\hat{R}_\pi$ and R_π would be uncorrelated, and the best we could do in terms of a linear fit would be $\hat{R}_\pi = Z$ (slope and intercept 0) — that is, we ignore R_π and simply model $\hat{R}_\pi$ as noise. The results summarized in Figure indicate that we are much closer to the former case than the latter. Over the 1000 random policies π that we generated, the correlation between $\hat{R}_\pi$ and R_π was positive and rejected the null hypothesis that the variables are uncorrelated well below the 0.01 level of significance; furthermore, the least squares linear fit gave a slope coefficient close to 1.0 and a y-intercept close to 0, as predicted by the idealized case above.

Conclusion

In this paper we presented a detailed methodology for using RL in the design of a spoken dialogue system. We built a large dialogue system using our methodology, and showed that RL is able to effectively search a very large space of dialogue policies (2^{42} in size) using a relatively small amount of training dialogue data (311 dialogues from 54 subjects). Our learned policy outperformed not only our training policy, but also many standard dialogue policies from the literature. We also reported on analyses verifying that the learned MDP is a reasonable model of the user population's interaction with NJFun. As future work, we would like to at least partially automate the choice of the state features used in constructing the MDP, explore the use of richer POMDP models, and do additional empirical evaluation of the RL approach.

Acknowledgements The authors thank Fan Jiang for his substantial effort in implementing our NJFun system, Esther Levin and Roberto Pieraccini for help in using their DMD programming language, Weiland Eckert for maintaining the CTmedia platform, Mazin Rahim for help with Watson, and David McAllester, Richard Sutton, Esther Levin and Roberto Pieraccini for numerous helpful conversations on dialogue system design.

References

A. W. Biermann and Philip M. Long. 1996. The composition of messages in speech-graphics interactive systems. In *Proc. of the 1996 International Symposium on Spoken Dialogue*, pages 97–100.

R. Crites and A. Barto. 1996. Improving elevator performance using reinforcement learning. In *Proc. NIPS 8* pages 1017-1023.

M. Danieli and E. Gerbino. 1995. Metrics for evaluating dialogue strategies in a spoken language system. In *Proc. of the 1995 AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, pages 34–39.

S. Haller and S. McRoy, eds. 1998. Special Issue: Computational Models of Mixed-Initiative Interaction (Part I) User Modeling and User-Adapted Interaction: An international journal, Vol. 8, Nos. 3–4.

S. Haller and S. McRoy, eds. 1999. Special Issue: Computational Models of Mixed-Initiative Interaction (Part II) User Modeling and User-Adapted Interaction: An international journal, Vol. 9, Nos. 1–2.

L.P. Kaelbling and M.L. Littman and A.W. Moore 1996. Reinforcement Learning: A survey. In *Journal of Artificial Intelligence Research 4*, pages 237–285.

E. Levin, R. Pieraccini, and W. Eckert. 1997. Learning dialogue strategies within the Markov decision process framework. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*.

D. J. Litman, M. S. Kearns, S. Singh, and M. A. Walker. 2000. Automatic Optimization of Dialogue Management. In *Proc. of COLING 2000*.

S. Singh, M. S. Kearns, D. J. Litman, and M. A. Walker. 1999. Reinforcement learning for spoken dialogue systems. In *Proc. NIPS99*.

R. W. Smith 1998. An Evaluation of Strategies for Selectively Verifying Utterance Meanings in Spoken Natural Language Dialog. In *International Journal of Human-Computer Studies*, 48, pages 627–647.

R. S. Sutton. 1991. Planning by incremental dynamic programming. In *Proc. Ninth Conference on Machine Learning*, pages 353–357. Morgan-Kaufmann.

G.J. Tesauro. 1995. Temporal difference learning and TD-Gammon. In *Comm. ACM 38*, pages 58–68.

M. A. Walker, J. C. Fromer, and S. Narayanan. 1998. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *Proc. of COLING/ACL 98*, pages 1345–1352.