Automatic Optimization of Dialogue Management

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

Abstract

Designing the dialogue strategy of a spoken dialogue system involves many nontrivial choices. This paper presents a reinforcement learning approach for automatically optimizing dialogue strategy. We first present a practical methodology that addresses the technical challenges in applying reinforcement learning to a working dialogue system with human users. We then demonstrate how we have used this methodology to measurably improve performance in a large-scale experimental system.

1 Introduction

Recent advances in spoken language understanding have made it possible to develop dialogue systems for many applications. The role of the dialogue manager in such systems is to interact in a natural way to help the user complete the tasks that the system is designed to support. Typically, an expert designs a dialogue manager by hand, and has to make many nontrivial design choices that can seriously impact system performance. This paper presents a reinforcement learning (RL) methodology for *automatically* learning design choices that optimize system performance with respect to a chosen performance measure.

Consider the spoken dialogue system named NJFun, which we built to help users find fun places to go in New Jersey. A sample dialogue with NJFun is shown in Figure 1, with system utterances labeled Si and user utterances labeled Ui. When NJFun greets the user, it can provide an open greeting such as utterance S1, expecting that the user will take the *initiative*. Alternatively, it can provide a directive greeting in which the system takes the initiative, as in "Welcome to NJFun. Please say an activity name or say 'list activities' for a list of activities I know about." After the user's response U1, NJFun must decide whether it should explicitly *confirm* its understanding, as in utterances S2 and S3.

S1:	Welcome to NJFun. How may I help you?
U1:	I'd like to find um winetasting in Lambertville in the morning. (*ASR output: I'd like to find out wineries the in the Lambertville in the morning.*)
S2:	Did you say you are interested in Lambertville?
U2:	Yes.
S3:	Did you say you want to go in the morning?
U3:	Yes.
S4:	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'.
U4:	Good.

Figure 1: An example dialogue with NJFun.

NJFun can also simply continue on with the dialogue, as when it does not explicitly confirm that the user wants to find out about wineries. In NJFun, as shown in more detail below, decisions about initiative and confirmation strategies alone result in a search space of 2^{42} potential global dialogue strategies. Furthermore, the performance of a dialogue strategy depends on many other factors, such as the user population, the robustness of system components like the automatic speech recognizer (ASR), and task difficulty (Kamm et al., 1998; Danieli and Gerbino, 1995).

In the main, previous research has treated the specification of the dialogue management strategy as an iterative design problem: several versions of a system are created, dialogue corpora are collected with human users interacting with different versions of the system, a number of evaluation metrics are collected for each dialogue, and the different versions are statistically compared (Danieli and Gerbino, 1995; Sanderman et al., 1998). Due to experimental constraints, only a few global strategies can be explored in any one experiment.

However, recent work has suggested that dialogue strategy can be designed using the formalism of Markov decision processes (MDPs) and the algorithms of RL (Biermann and Long, 1996; Levin et al., 1997; Walker et al., 1998; Singh et al., 1999). More specifically, the MDP formalism suggests a method for optimizing dialogue strategies from sample dialogue data. The main advantage of this approach is the potential for computing an optimal dialogue strategy within a much larger search space, using a relatively small number of training dialogues.

This paper presents a large-scale application of RL to the problem of optimizing dialogue strategy selection in the NJFun system, and experimentally demonstrates the utility of the approach. Section 2 explains how we apply RL to dialogue systems, then Section 3 describes the NJFun system in detail. Section 4 describes how NJFun optimizes its dialogue strategy from experimentally obtained dialogue data. Section 5 reports results from testing the learned strategy demonstrating that our approach improves task completion rates (our chosen measure for performance optimization). A companion paper provides only an abbreviated system and dialogue manager description, but includes additional results not presented here (Anonymous, 2000), such as analysis establishing the veracity of the MDP we learn, and comparisons of our learned strategy to strategies hand-picked by dialogue experts.

2 Reinforcement Learning for Dialogue

Due to space limitations, we present only a brief overview of how dialogue strategy optimization can be viewed as a RL problem; for a more detailed exposition, see Singh et al. (1999), Walker et al. (1998), Levin et al. (1997). A dialogue strategy is a mapping from a set of *states* (which summarize the entire dialogue so far) to a set of *actions* (such as the system's utterances and database queries). There are multiple reasonable action choices in each state; typically these choices are made by the system designer. Our RL-based approach is to build a system that explores these choices in a systematic way through experiments with representative human users. A scalar performance measure, called a reward, is then calculated for each experimental dialogue. (We discuss various choices for this reward measure later, but typically only terminal dialogue states have nonzero rewards, and the reward measures are quantities directly obtainable from the experimental set-up, such as user satisfaction or task completion for the dialogue.) This experimental data is used to construct a MDP which models the users' interaction with the system. The problem of learning the best dialogue strategy from data is thus reduced to computing the optimal *policy* for choosing actions in an MDP — that is, the system's goal is to take actions so as to maximize expected reward. The computation of the optimal policy *given* the MDP can be done efficiently using standard RL algorithms.

How do we build the desired MDP from sample dialogues? Following (Singh et al., 1999), we can view a dialogue as a trajectory in the chosen state space determined by the system actions and user responses:

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

Here $s_i \rightarrow_{a_i,r_i} s_{i+1}$ 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}. Dialogue sequences obtained from training data can be used to empirically estimate the transition probabilities $P(s'|s,a)$ (denoting the probability of a transition to state s', given that the system was in state s and took action a), and the reward function $R(s,a)$. The estimated transition probabilities and reward function constitute an MDP model of the user population's interaction with the system.

Given this MDP, the expected cumulative reward (or *Q-value*) $Q(s,a)$ of taking action a from state s can be calculated in terms of the Q-values of successor states via the following recursive equation (Sutton, 1991):

$$Q(s,a) = R(s,a) + \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a').$$

These Q-values can be estimated to within a desired threshold using the standard RL *value iteration* algorithm, which iteratively updates the estimate of $Q(s,a)$ based on the current Q-values of neighboring states. Once value iteration is completed, the optimal dialogue strategy (according to our estimated model) is obtained by selecting the action with the maximum Q-value at each dialogue state.

While this approach is theoretically appealing, the difficulty of obtaining sample human dialogues makes it crucial to limit the size of the state space, in order to minimize data sparsity problems, while still retaining enough information in the state to learn an accurate model. Our approach is to work directly in a minimal but carefully designed state space (Singh et al., 1999).

The contribution of this paper is to develop and empirically validate a practical methodology for us-

Action	Prompt
GreetS	Welcome to NJFun. Please say an activity name or say 'list activities' for a list of activities I know about.
GreetU	Welcome to NJFun. How may I help you?
ReAsk1S	I know about amusement parks, aquariums, cruises, historic sites, museums, parks, theaters, wineries, and zoos. Please say an activity name from this list.
ReAsk1M	Please tell me the activity type.You can also tell me the location and time.
Ask2S	Please say the name of the town or city that you are interested in.
Ask2U	Please give me more information.
ReAsk2S	Please tell me the name of the town or city that you are interested in.
ReAsk2M	Please tell me the location that you are interested in. You can also tell me the time.

Figure 2: Sample initiative strategy choices.

ing RL to *build* a real dialogue system that optimizes its behavior from dialogue data. Our methodology involves 1) representing a dialogue strategy as a mapping from each state in the chosen state space $\mathcal{S}$ to a set of dialogue actions, 2) deploying an initial training system that generates *exploratory* training data with respect to $\mathcal{S}$, 3) constructing an MDP model from the obtained training data, 4) using value iteration to learn the optimal dialogue strategy in the learned MDP, and 4) redeploying the system using the learned state/action mapping. The next section details the use of this methodology to design the NJFun system.

3 The NJFun System

NJFun is a real-time spoken dialogue system that provides users with information about things to do in New Jersey. NJFun is built using a general purpose platform for spoken dialogue systems (Levin et al., 1999), with support for modules for automatic speech recognition (ASR), spoken language understanding, text-to-speech (TTS), database access, and dialogue management. NJFun uses a speech recognizer with stochastic language and understanding models trained from example user utterances, and a TTS system based on concatenative diphone synthesis. Its database is populated from the `nj.online` webpage to contain information about activities. NJFun indexes this database using three attributes: activity type, location, and time of day (which can assume values morning, afternoon, or evening).

Informally, the NJFun dialogue manager sequentially queries the user regarding the activity, location and time attributes, respectively. NJFun first asks the user for the current attribute (and possibly the other attributes, depending on the initiative). If the current attribute's value is not obtained, NJFun asks for the attribute (and possibly the later attributes) again. If NJFun still does not obtain a value, NJFun moves on to the next attribute(s). Whenever NJFun successfully obtains a value, it can confirm the value, or move on and attempt to obtain the next attribute(s). When NJFun has finished asking about the attributes, it queries the database (using a wildcard for each unobtained attribute value). Note that the length of NJFun dialogues ranges from 1 to 12 user utterances before the database query.

As discussed above, our methodology for using RL to optimize dialogue strategy requires that all potential actions for each state be specified. Note that at some states it is easy for a human to make the correct action choice. We made obvious dialogue strategy choices in advance, and used learning only to optimize the difficult choices (Walker et al., 1998). In NJFun, we restricted the action choices to 1) the type of initiative to use when asking or reasking for an attribute, and 2) whether to confirm an attribute value once obtained. The optimal actions may vary with dialogue state, and are subject to active debate in the literature.

The examples in Figure 2 shows that NJFun can ask the user about the first 2 attributes[1] using three types of initiative, based on the combination of the wording of the system prompt (*open* versus *directive*), and the type of grammar NJFun uses during ASR (*restrictive* versus *non-restrictive*). If NJFun uses an open question with an unrestricted grammar, it is using *user initiative* (e.g., GreetU). If NJFun instead uses a directive prompt with a restricted grammar, the system is using *system initiative* (e.g., GreetS). If NJFun uses a directive question with a non-restrictive grammar, it is using *mixed initiative*, because it is giving the user an opportunity to take the initiative by supplying extra information (e.g.,

[1] "Greet" is equivalent to asking for the first attribute. NJFun always uses system initiative for the third attribute, because at that point the user can only provide the time of day.

ReAsk1M).

NJFun can also vary the strategy used to confirm each attribute. If NJFun asks the user to explicitly verify an attribute, it is using *explicit confirmation* (e.g., ExpConf2 for the location, exemplified by S2 in Figure 1). If NJFun does not generate any confirmation prompt, it is using *no confirmation* (an action we call NoConf).

Solely for the purposes of controlling its operation (as opposed to the learning, which we discuss in a moment), NJFun internally maintains an *operations vector* of 14 variables. 2 variables track whether the system has greeted the user, and which attribute the system is currently attempting to obtain. For each of the 3 attributes, 4 variables track whether the system has obtained the attribute's value, the system's confidence in the value (if obtained), the number of times the system has asked the user about the attribute, and the type of ASR grammar most recently used to ask for the attribute.

The formal state space $\mathcal{S}$ maintained by NJFun for the purposes of learning is much simpler than the operations vector, due to the data sparsity concerns already discussed. The dialogue state space $\mathcal{S}$ contains only 7 variables, which are summarized in Figure 3, and is easily computed from the operations vector. The "greet" variable tracks whether the system has greeted the user or not (no=0, yes=1). "Attr" specifies which attribute NJFun is currently attempting to obtain or verify (activity=1, location=2, time=3, done with attributes=4). "Conf" represents the confidence that NJFun has after obtaining a value for an attribute. The values 0, 1, and 2 represent low, medium and high ASR confidence.[2] The values 3 and 4 are set when ASR hears "yes" or "no" after a confirmation question. "Val" tracks whether NJFun has obtained a value for the attribute (no=0, yes=1). "Times" tracks the number of times that NJFun has asked the user about the attribute. "Gram" tracks the type of grammar most recently used to obtain the attribute (0=non-restrictive, 1=restrictive). Finally, "history" represents whether NJFun had trouble understanding the user in the earlier part of the conversation (bad=0, good=1). We omit the full definition, but as an example, when NJFun is working on the second attribute (location), the history variable is set to 0 if NJFun does not have an activity, has an activity but has no confidence in the value, or needed two queries to obtain the activity.

greet	attr	conf	val	times	gram	history
0,1	1,2,3,4	0,1,2,3,4	0,1	0,1,2	0,1	0,1

Figure 3: State features and values.

As mentioned above, the goal is to design a small state space that makes enough critical distinctions to support learning. The use of $\mathcal{S}$ reduces the number of states to only 62,[3] and supports the construction of an MDP model that is not sparse with respect to $\mathcal{S}$, even using limited training data. The state space that we utilize here, although minimal, allows us to make initiative decisions based on the success of earlier exchanges, and confirmation decisions based on ASR confidence scores and grammars.

The state/action mapping representing NJFun's initial dialogue strategy EIC (Exploratory for Initiative and Confirmation) is given in Figure 4. Only the exploratory portion of the strategy is shown, namely all those states for which NJFun has an action choice. For each such state, we list the two choices of actions available. (The action choices in boldface are the ones eventually identified as optimal by the learning process, and are discussed in detail later.) The EIC strategy chooses *randomly* between these two actions when in the indicated state, in order to maximize exploration and minimize data sparseness when constructing our model. Since there are 42 states with 2 choices each, there is a search space of 2^{42} potential dialogue strategies; the goal of the RL is to identify an apparently optimal strategy from this large search space. Note that due to the randomization of the EIC strategy, the prompts are designed to ensure the coherence of all possible action sequences.

Figure 5 illustrates how the dialogue strategy in Figure 4 generates the dialogue in Figure 1. Each row indicates the state that NJFun is in, the action executed in this state, the corresponding turn in Figure 1, and the reward received. The initial state represents that NJFun will first attempt to obtain attribute 1. NJFun executes GreetU (although as shown in Figure 4, GreetS is also possible), generating the first utterance in Figure 1. After the user's response, the next state represents that NJFun has now greeted the user and obtained the ac-

[2]For each utterance, the ASR output includes not only the recognized string, but also an associated acoustic confidence score. Based on data obtained during system development, we defined a mapping from raw confidence values into 3 approximately equally populated partitions.

[3]This number refers to those states that can actually occur in a dialogue.

State							Action Choices
g	a	c	v	t	g	h	
0	1	0	0	0	0	0	GreetS,**GreetU**
1	1	0	0	1	0	0	**ReAsk1S**,ReAsk1M
1	1	0	1	0	0	0	NoConf,**ExpConf1**
1	1	0	1	0	1	0	NoConf,**ExpConf1**
1	1	1	1	0	0	0	NoConf,**ExpConf1**
1	1	1	1	0	1	0	NoConf,**ExpConf1**
1	1	2	1	0	0	0	**NoConf**,ExpConf1
1	1	2	1	0	1	0	**NoConf**,ExpConf1
1	1	4	0	0	0	0	ReAsk1S,ReAsk1M
1	1	4	0	1	0	0	**ReAsk1S**,ReAsk1M
1	2	0	0	0	0	0	Ask2S,**Ask2U**
1	2	0	0	0	0	1	Ask2S,**Ask2U**
1	2	0	0	1	0	0	ReAsk2S,**ReAsk2M**
1	2	0	0	1	0	1	ReAsk2S,**ReAsk2M**
1	2	0	1	0	0	0	NoConf,**ExpConf2**
1	2	0	1	0	0	1	NoConf,**ExpConf2**
1	2	0	1	0	1	0	NoConf,**ExpConf2**
1	2	0	1	0	1	1	NoConf,ExpConf2
1	2	1	1	0	0	0	NoConf,**ExpConf2**
1	2	1	1	0	0	1	NoConf,**ExpConf2**
1	2	1	1	0	1	0	**NoConf**,ExpConf2
1	2	1	1	0	1	1	NoConf,**ExpConf2**
1	2	2	1	0	0	0	**NoConf**,ExpConf2
1	2	2	1	0	0	1	NoConf,**ExpConf2**
1	2	2	1	0	1	0	**NoConf**,ExpConf2
1	2	2	1	0	1	1	**NoConf**,ExpConf2
1	2	4	0	0	0	0	**ReAsk2S**,ReAsk2M
1	2	4	0	0	0	1	ReAsk2S,ReAsk2M
1	2	4	0	1	0	0	ReAsk2S,**ReAsk2M**
1	2	4	0	1	0	1	ReAsk2S,**ReAsk2M**
1	3	0	1	0	0	0	NoConf,ExpConf3
1	3	0	1	0	0	1	NoConf,ExpConf3
1	3	0	1	0	1	0	**NoConf**,ExpConf3
1	3	0	1	0	1	1	NoConf,**ExpConf3**
1	3	1	1	0	0	0	NoConf,**ExpConf3**
1	3	1	1	0	0	1	NoConf,ExpConf3
1	3	1	1	0	1	0	**NoConf**,ExpConf3
1	3	1	1	0	1	1	NoConf,**ExpConf3**
1	3	2	1	0	0	0	NoConf,**ExpConf3**
1	3	2	1	0	0	1	NoConf,ExpConf3
1	3	2	1	0	1	0	**NoConf**,ExpConf3
1	3	2	1	0	1	1	NoConf,**ExpConf3**

Figure 4: Exploratory portion of EIC strategy.

State g a c v t g h	Action	Turn	Reward
0 1 0 0 0 0 0	GreetU	S1	0
1 1 2 1 0 0 0	NoConf	-	0
1 2 2 1 0 0 1	ExpConf2	S2	0
1 3 2 1 0 0 1	ExpConf3	S3	0
1 4 0 0 0 0 0	Tell	S4	1

Figure 5: Generating the dialogue in Figure 1.

tivity value with high confidence, by using a non-restrictive grammar. NJFun chooses not to confirm the activity, which causes the state to change but no prompt to be generated. The third state represents that NJFun is now working on the second attribute (location), that it already has this value with high confidence (location was obtained with activity after the user's first utterance), and that the dialogue history is good.[4] This time NJFun chooses to confirm the attribute with the second NJFun utterance, and the state changes again. The processing of time is similar to that of location, which leads NJFun to the final state, where it performs the action "Tell" (corresponding to querying the database, presenting the results to the user, and asking the user to provide a reward). Note that in NJFun, the reward is always 0 except at the terminal state, as shown in the last column of Figure 5.

4 Experimentally Optimizing a Strategy

We collected experimental dialogues for both training and testing our system. To obtain training dialogues, we implemented NJFun using the EIC dialogue strategy described in Section 3. We used these dialogues to build an empirical MDP, and then computed the optimal dialogue strategy in this MDP (as described in Section 2). In this section we describe our experimental design and the learned dialogue strategy. In the next section we present results from testing our learned strategy and show that it improves task completion rates, the performance measure we chose to optimize.

Subjects in our experiment were employees not associated with the NJFun project. 54 subjects were used for training, while 21 subjects were used for testing. Subjects were distributed such that the training and testing pools were comparable with respect to gender, English as a first language, and expertise with spoken dialogue systems.

During both training and testing, subjects carried out free-form conversations with NJFun to complete six application tasks. For example, the user in Figure 1 was executing the following task: "You feel thirsty and want to do some winetasting in the morning. Are there any wineries close by your house in Lambertville?" Subjects read each task description on a web page, then called NJFun from their office phone. At the end of the task, NJFun asked them to verbally provide feedback on their experience (e.g., utterance S4 in Figure 1). Users then hung up the phone and filled out a user survey on the web.

The training phase of the experiment resulted in 311 complete dialogues (not all subjects completed all tasks), for which NJFun logged the sequence

[4] Recall that only the current attribute's features are in the state. However, the operations vector contains information regarding previous attributes.

of states and the corresponding executed actions. The number of samples per state for the initial ask choices are:

0 1 0 0 0 0 0	GreetS=155	GreetU=156
1 2 0 0 0 0 0	Ask2S=93	Ask2U=72
1 2 0 0 0 0 1	Ask2S=36	Ask2U=48

Such data illustrates that our use of a random action choice strategy led to a fairly balanced action distribution per state. Similarly, our small state space, and the fact that we only allowed 2 action choices per state, prevented a data sparseness problem. The first state in Figure 4 was the most frequently visited state (with 311 visits), as it was the initial state for every dialogue. Only 8 states were visited less than 10 times, and were states that only occur near the end of a dialogue.

The logged data was then used to construct the empirical MDP. As we have mentioned, the measure we chose to optimize is a binary reward function based on the strongest possible measure of task completion, called **StrongComp**, that takes on value 1 if NJFun queries the database using exactly the attributes specified in the task description, and 0 otherwise. Then we computed the optimal dialogue strategy in this MDP using RL (cf. Section 2). The action choices constituting the learned strategy are in boldface in Figure 4. Note that no choice was fixed for several states, meaning that the Q-values were identical after value iteration. Thus, even when using the learned strategy, NJFun still sometimes chooses randomly between certain action pairs.

Intuitively, the learned strategy says that the optimal use of initiative is to begin with user initiative, then back off to either mixed or system initiative when reasking for an attribute. Note, however, that the specific backoff method differs with attribute (e.g., system initiative for attribute 1, but generally mixed initiative for attribute 2). With respect to confirmation, the optimal strategy is to mainly confirm at the lower confidence values. Again, however, the point at which confirmation becomes unnecessary differs across attributes (e.g., confidence level 2 for attribute 1, but sometimes lower levels for attributes 2 and 3). NJFun can learn such fine-grained distinctions because the optimal strategy is in effect based on a comparison of 2^{42} possible exploratory strategies. Both the initiative and confirmation results suggest that the initial portion of the dialogue was the most problematic for NJFun. Figure 1 is an example dialogue obtained using the optimal strategy.

5 Experimentally Evaluating the Strategy

For the testing phase, NJFun was reimplemented to use the learned strategy. 21 test subjects then performed the same 6 tasks used during training, resulting in 124 complete test dialogues. One of our main results is that task completion as measured by StrongComp increased from 52% in training to 64% in testing ($p < .06$)[5].

There is also a significant interaction effect between strategy and task (p<.01) for StrongComp. Previous work has suggested that novice users perform comparably to experts after only 2 tasks (Kamm et al., 1998). Since our learned strategy was based on 6 tasks with each user, one explanation of the interaction effect is that the learned strategy is slightly optimized for expert users. To explore this hypothesis, we divided our corpus into dialogues with "novice" (tasks 1 and 2) and "expert" (tasks 3-6) users. We found that the learned strategy did in fact lead to a large and significant improvement in StrongComp for experts (EIC=.46, learned=.69, p<.001), and a non-significant degradation for novices (EIC=.66, learned=.55, p<.3).

An apparent limitation of these results is that EIC may not be the best baseline strategy for comparison to our learned strategy. A more standard alternative would be comparison to the very best hand-designed fixed strategy. However, there is no agreement in the literature, nor amongst the authors, as to what the best hand-designed strategy might have been. There is agreement, however, that the best strategy is sensitive to many unknown and unmodeled factors: the user population, the specifics of the task, the particular ASR used, etc. Furthermore, EIC was carefully designed so that the random choices it makes never results in an unnatural dialogue. Finally, a companion paper (Anonymous, 2000) shows that the performance of the learned strategy is better than several "standard" fixed strategies (such as global system-initiative and no-confirmation).

We also investigated the performance of the learned strategy on a number of other reward measures for which it was not explicitly optimized.

[5]The experimental design described above consists of 2 factors: the within-in group factor *strategy* and the between-groups factor *task*. We use a two-way analysis of variance (ANOVA) to compute whether main and interaction effects of strategy are statistically significant (p<.05) or indicative of a statistical trend (p < .10). *Main effects* of strategy are task-independent, while *interaction* effects involving strategy are task-dependent.

Measure	EIC (n=311)	Learned (n=124)	p
WeakComp	1.75	2.19	.02
StrongComp	0.52	0.64	.06
ASR	2.50	2.67	.04
Feedback	0.18	0.11	.42
UserSat	13.38	13.29	.86

Table 1: Main effects of dialogue strategy.

WeakComp is a relaxed version of task completion that gives partial credit: if all attribute values are either correct or wildcards, the value is the sum of the correct number of attributes. Otherwise, at least one attribute is wrong (e.g., the user says "Lambertville" but the system hears "Morristown"), and the value is -1. **ASR** approximates speech recognition accuracy for the database query, and is computed by adding 1 for each correct attribute value and .5 for every wildcard. Thus, if the task is to go winetasting near Lambertville in the morning, and the system queries the database for an activity in New Jersey in the morning, StrongComp=0, WeakComp=1, and ASR=2. In addition to the objective measures discussed above, we also computed two *subjective* measures. **Feedback** is obtained from the dialogue (e.g. S4 in Figure 5), by mapping *good, so-so, bad* to 1, 0, and -1, respectively. User satisfaction (**UserSat**, ranging from 0-20) is obtained by summing the answers of the web-based user survey.

Table 1 summarizes the difference in performance of NJFun for all our reward functions from training (EIC) to test (learned strategy for StrongComp). For WeakComp, the average reward increased from 1.75 to 2.19 ($p < 0.02$), while for ASR the average reward increased from 2.5 to 2.67 ($p < 0.04$). We note that these improvements occur even though the learned strategy was not optimized for these measures.

The last two rows of the table show that for the subjective measures, performance does not significantly differ for the EIC and learned strategies. Interestingly, the distributions of the subjective measures move to the middle from training to testing, i.e., test users reply to the survey using less extreme answers than training users. Explaining the subjective results is an area for future work.

6 Discussion

This paper presents a practical methodology for applying RL to optimizing dialogue strategies in spoken dialogue systems, and demonstrates empirically that it improved performance over the EIC policy in NJFun. A companion paper (Anonymous, 2000) shows that our strategy is not only better than EIC, but also than other fixed choices proposed in the literature. While RL has been applied in previous work on spoken dialogue systems, we report the first large-scale application of RL to a dialogue system that interacts in real time with human users. This is in contrast to previous work on using RL for dialogue system design that either proposed it without implementation (Biermann and Long, 1996), or that used a simulated user population in their implementation (Levin et al., 1997), or that was limited to exploring a small search space consisting of 18 policies (Walker et al., 1998) (as compared to the 2^{42} policies explored here). We also note that our learned strategy varied initiative and confirmation decisions at a finer grain than previous work, and as such is not a standard policy investigated in the dialogue system literature. In particular, we would not have predicted the complex and interesting back-off strategy with respect to initiative when reasking for an attribute.

As future work we would like to understand the aforementioned results on the subjective reward measures, explore the potential difference between optimizing for expert users and novices, automate the choice of state space for dialogue systems, investigate the use of a learned reward function (as in (Walker et al., 1998)), and explore the use of more informative non-terminal rewards.

References

Anonymous. 2000. Manuscript submitted to *AAAI*.

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.

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.

C. A. Kamm, D. J. Litman, and M. A. Walker. 1998. From novice to expert: The effect of tutorials on user expertise with spoken dialogue systems. In *Proc. of the International Conference on Spoken Language Processing, ICSLP98*.

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*.

E. Levin, R. Pieraccini, W. Eckert, G. Di Fabbrizio, and S. Narayanan. 1999. Spoken language dialogue: From theory to practice. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding, ASRUU99*.

A. Sanderman, J., E. den Os, L. Boves, and A. Cremers. 1998. Evaluation of the Dutchtrain timetable information system developed in the ARISE project. In *Interactive Voice Technology for Telecommunications Applications, IVTTA*, pages 91–96.

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

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

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.