NJFun: A Reinforcement Learning Spoken Dialogue System

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

Abstract

This paper describes NJFun, a real-time spoken dialogue system that provides users with information about things to do in New Jersey. NJFun automatically optimizes its dialogue strategy over time, by using a methodology for applying reinforcement learning to a working dialogue system with human users.

1 Introduction

Using the formalism of Markov decision processes (MDPs) and the algorithms of reinforcement learning (RL) has become a standard approach to many AI problems that involve an agent learning to optimize reward by interaction with its environment (Sutton and Barto, 1998). We have adapted the methods of RL to the problem of automatically learning a good dialogue strategy in a fielded spoken dialogue system. Here is a summary of our proposed methodology for developing and evaluating spoken dialogue systems using RL:

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

In this demonstration session paper, we briefly describe our system, present some sample dialogues, and summarize our main contributions and limitations. Full details of our work (e.g. our reinforcement learning methodology, analysis establishing the veracity of the MDP we learn, a description of an experimental evaluation of NJFun, analysis of our learned dialogue strategy) can be found in two forthcoming technical papers (Singh et al., 2000; Litman et al., 2000).

2 The NJFun System

NJFun is a real-time spoken dialogue system that provides users with information about things to do in New Jersey.[1] An example dialogue with NJFun is shown in Figure 1. NJFun is built using an internal platform for spoken dialogue systems. NJFun uses a speech recognizer with stochastic language 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.

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).[2] When NJFun has finished asking about the attributes, it queries the database (using a wildcard for each unobtained attribute value).

We use reinforcement learning (RL) to optimize dialogue strategy. RL 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. In NJFun, we restricted the action choices to 1) the type of initiative to use

[1] This system description section is taken from (Litman et al., 2000).

[2] Note that it is possible for users to specify multiple attributes, in any order, in a single utterance. However, NJFun will always process multiple attributes using its predefined sequential ordering.

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 (after optimization via RL).

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.

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[3] 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., 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 data sparsity concerns. 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. 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.

In order to apply RL with a limited amount of training data, we need to design a small state space

[3] "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.

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.

that makes enough critical distinctions to support learning. The use of $\mathcal{S}$ yields a state space of size 62. 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.

In order to learn a good dialogue strategy via RL we have to explore the state action space. The state/action mapping representing NJFun's initial exploratory 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.) 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 activity 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. 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.

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							Action	Turn	Reward
g	a	c	v	t	g	h			
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.

3 Sample Dialogues

Figures 1 and 6 present two dialogues that were generated using different dialogue strategies in the EIC search space. For example, although NJFun begins both dialogues in the first state in Figure 4, NJFun executes GreetU in one dialogue and GreetS in the other. Similarly, although the next state of NJFun

S1: Welcome to NJFun. Please say an activity name or say 'list activities' for a list of activities I know about.
U1: I would like to visit a historic site in Stanhope this morning.
(*ASR output: I would like to the zoo historic sites in stanhope historic*)
S2: Did you say you are interested in going to a zoo?
U2: No.
S3: I know about amusement parks, aquariums, cruises, historic sites, museums, parks, theaters, wineries, and zoos. Please say an activity name from this list.
U3: I would like to visit a historic site. (*ASR output: I would like to visit historic sites*)
S4: Did you say you are interested in going to a historic site?
U4: Yes.
S5: Please give me more information.
U5: *(user says nothing)*
S6: Please tell me the location that you are interested in. You can also tell me the time.
U6: Visit a historic. (*ASR output: What historic*)
S7: What time of the day do you want to go?
U7: Stanhope. (*no ASR output*)
S8: Do you want to go in the morning, in the afternoon, or in the evening?
U8: Morning.
S9: Did you say you want to go in the morning?
U9: Yes.
S10: I found 11 historic sites in New Jersey that are open in the morning. The first 3 [...] Would you like to hear more?
U10: No.
S11: Thank you for using the system. Please give me feedback by saying 'good', 'so-so', or 'bad'.
U11: Bad.

Figure 6: An example training dialogue with NJFun.

is the same in both dialogues ("1 1 2 1 0 0 0"), the activity is not confirmed in the first dialogue.

4 Contributions

The main contribution of this work is that we have developed and empirically validated a practical methodology for using RL to *build* a real dialogue system that optimizes its behavior from dialogue data. Unlike traditional approaches to learning dialogue strategy from data, which are limited to searching a handful of policies, our RL approach is able to search many tens of thousands of dialogue strategies. In particular, the traditional approach is to pick a handful of strategies that experts intuitively feel are good, implement each policy as a separate system, collect data from representative human users for each system, and then use standard statistical tests on that data to pick the best system, e.g. (Danieli and Gerbino, 1995). In contrast, our use of RL allowed us to explore 2^{42} strategies that were left in our search space after we excluded strategies that were clearly suboptimal.

An empirical validation of our approach is detailed in two forthcoming technical papers (Singh et al., 2000; Litman et al., 2000). We obtained 311 dialogues with the exploratory (i.e., training) version of NJFun, constructed an MDP from this training data, used RL to compute the optimal dialogue strategy in this MDP, reimplemented NJFun such that it used this learned dialogue strategy, and obtained 124 more dialogues. Our main result was that task completion improved from 52% to 64% from training to test data. Furthermore, analysis of our MDP showed that the learned strategy was not only better than EIC, but also better than other fixed choices proposed in the literature (Singh et al., 2000).

5 Limitations

The main limitation of this effort to automate the design of a good dialogue strategy is that our current framework has nothing to say about how to choose the reward measure, or how to best represent dialogue state. In NJFun we carefully but manually designed the state space of the dialogue. In the future, we hope to develop a learning methodology to automate the choice of state space for dialogue systems. With respect to the reward function, our empirical evaluation investigated the impact of using a number of reward measures (e.g., user feedback such as U4 in Figure 1, task completion rate, ASR accuracy), and found that some rewards worked better than others. We would like to better understand these differences among the reward measures, investigate the use of a learned reward function, and explore the use of non-terminal rewards.

References

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

D. Litman, M. Kearns, S. Singh, and M. Walker. 2000. Automatic optimization of dialogue management. Manuscript submitted for publication.

S. Singh, M. Kearns, D. Litman, and M. Walker. 2000. Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proceedings of AAAI 2000.*

R. S. Sutton and A. G. Barto. 1998. *Reinforcement Learning: An Introduction.* MIT Press.