Learning optimal dialogue management rules by using reinforcement learning and inductive logic programming

Renaud Lecœuche
Microsoft Corporation
Compass House, 80-82 Newmarket road
Cambridge CB5 8DZ, United Kingdom
renaudle@microsoft.com

Abstract

Developing dialogue systems is a complex process. In particular, designing efficient dialogue management strategies is often difficult as there are no precise guidelines to develop them and no sure test to validate them. Several suggestions have been made recently to use reinforcement learning to search for the optimal management strategy for specific dialogue situations. These approaches have produced interesting results, including applications involving real world dialogue systems. However, reinforcement learning suffers from the fact that it is state based. In other words, the optimal strategy is expressed as a decision table specifying which action to take in each specific state. It is therefore difficult to see whether there is any generality across states. This limits the analysis of the optimal strategy and its potential for re-use in other dialogue situations. In this paper we tackle this problem by learning rules that generalize the state-based strategy. These rules are more readable than the underlying strategy and therefore easier to explain and re-use. We also investigate the capability of these rules in directing the search for the optimal strategy by looking for generalization whilst the search proceeds.

1 Introduction

As dialogue systems become ubiquitous, dialogue management strategies are receiving more and more attention. They define the system behavior and mainly determine how well or badly it is perceived by users. Generic methodologies exist for developing and testing management strategies. Many of these take a user-centric approach based on Wizard of Oz studies and iterative design (Bernsen et al., 1998). However there are still no precise guidelines about when to use specific techniques such as mixed-initiative. Reinforcement learning has been used in several recent approaches to search for the optimal dialogue management strategy for specific dialogue situations (Levin and Pieraccini, 1997; Litman et al., 2000; Singh et al., 2000; Walker, 2000). In these approaches, a dialogue is seen as a walk through a series of states, from an initial state when the dialogue begins until a terminal state when the dialogue ends. The actions of the dialogue manager as well as those of the user influence the transitions between states. Each transition is associated with a reward, which expresses how good or bad it was to make that transition. A dialogue strategy is then seen as a Markov Decision Process (Levin et al., 1998). Reinforcement learning can be used in this framework to search for an optimal strategy, i.e., a strategy that makes the expected sum of rewards maximal for all the training dialogues. The main idea behind reinforcement learning is to explore the space of possible dialogues and select the strategy which optimizes the expected rewards (Mitchell, 1997, ch. 13). Once the optimal strategy has been found, it can be implemented in the final system.

Reinforcement learning is state-based. It finds out which action to take next given the current state. This makes the explanation of the strategy relatively hard and limits its potential re-use to other dialogue situations. It is quite difficult to find out whether generic lessons can be learned from the optimal strategy. In this paper we use inductive logic programming (ILP) to learn sets of rules that generalize the optimal strategy. We show that these can be simpler to interpret than the decision tables given by reinforcement learning and can help modify and re-use the strategies. This is important as human dialogue designers are usually ultimately in

charge of writing and changing the strategies.

The paper is organized as follows. We first describe in section 2 a simple dialogue system which we use as an example throughout the paper. In section 3, we present our method and results on using ILP to generalize the optimal dialogue management strategy found by reinforcement learning. We also investigate the use of the rules learned during the search of the optimal strategy. We show that, in some cases, the number of dialogues needed to obtain the optimal strategy can be dramatically reduced. Section 4 presents our current results on this aspect. We compare our approach to other current pieces of work in section 5 and conclude the paper in section 6.

2 Example dialogue system

In this section we present a simple dialogue system that we use in the rest of the paper to describe and explain our results. This system will be used with automated users in order to simulate dialogues. The aim of the system is to be simple enough so that its operation is easy to understand while being complex enough to allow the study of the phenomena we are interested in. This will provide a simple way to explain our approach.

We chose a system whose goal is to find values for three pieces of information, called, unoriginally, a, b and c. In a practical system such as an automated travel agent for example, these values could be departure and arrival cities and the time of a flight.

We now describe the system in terms of states, transitions, actions and rewards, which are the basic notions of reinforcement learning. The system has four actions at its disposal: prepare to ask (prepAsk) a question about one of the pieces of information, prepare to recognize (prepRec) a user's utterance about a piece of information, ask and recognize (ask&recognize) which outputs all the prepared questions and tries to recognize all the expected utterances, and end (end) which terminates the dialogue. We chose these actions as they are common, in one form or another, in most speech dialogue systems. To get a specific piece of information, the system must prepare a question about it and expect a user utterance as an answer before carrying out an ask&recognize action. The system can try to get more than one piece of information in a single ask&recognize action by preparing more than one question and preparing to recognize more than one answer.

Actions are associated with rewards or penalties. Every system action, except ending, has a penalty of -5 corresponding to some imagined processing cost. Ending provides a reward of 100 times the number of pieces of information known when the dialogue ends. We hope that these numbers simulate a realistic reward function. They could be tuned to reflect user satisfaction for a real dialogue manager.

The state of the system represents which pieces of information are known or unknown and what questions and recognitions have been prepared. There is also a special end state. For this example, there are 513 different states.

Pieces of information become known when users answer the system's questions. In our tutorial example, we used automated users. These users always give one piece of information if properly asked as explained above, and answer potential further questions with a decreasing probability (0.5 for a second piece of information, and 0.25 for a third in our example). We could tune these probabilities to reflect real user behavior. Using simulated users enables us to quickly train our system. It could also allow us to test the usefulness of ILP under different conditions.

3 Learning rules from optimal strategy

In this section we explain how we obtain and interpret rules expressing the optimal management strategy found by reinforcement learning for the system presented in section 2 as well as a more realistic one.

3.1 Example system

We first search for the optimal strategy of our example system by using reinforcement learning. We do this by having repetitive dialogues with the automated users and evaluating the average reward of the actions taken by the system. When deciding what to do in each state, we choose the up-to-now best action with probability 0.8 and other actions with uniform probability totaling 0.2. This allows the system to explore the dialogue space while preferably following the best strategy found. The optimal

State	Action
{unknown(c), unknown(b), unknown(a)}	prepRec(a)
{unknown(c), unknown(b), unknown(a), prepRec(a)}	prepAsk(a)
{unknown(c), unknown(b), unknown(a), prepRec(a), prepAsk(a)}	ask&recognize
{unknown(c), unknown(b), known(a)}	prepRec(b)
{unknown(c), unknown(b), known(a), prepRec(b)}	prepAsk(b)
{unknown(c), unknown(b), known(a), prepRec(b), prepAsk(b)}	ask&recognize
{unknown(c), known(b), known(a)}	prepRec(c)
{unknown(c), known(b), known(a), prepRec(c)}	prepAsk(c)
{unknown(c), known(b), known(a), prepRec(c), prepAsk(c)}	ask&recognize
{known(c), known(b), known(a)}	end

Table 1: Tutorial example optimal strategy

strategy for the tutorial example is shown in table 1. This strategy is very simple: ask one piece of information at a time until all the pieces have been collected, and then end the dialogue. A typical dialogue following this strategy would simply go like this, using the travel agent example:

```
S: Where do you want to leave from?
U: Cambridge.
S: Where do you want to go to?
U: Seattle.
S: When do you want to travel?
U: Tomorrow.
```

Then, in order to learn rules generalizing the optimal strategy, we use FOIDL. FOIDL is a program which learns first-order rules from examples (Mooney and Califf, 1995; Mitchell, 1997, ch. 10). FOIDL starts with rules without conditions and then adds further terms so that they cover the examples given but not others. In our case, rule conditions are about properties of states and rule actions are the best actions to take. Some advantages of FOIDL are that it can learn from a relatively small set of positive examples without the need for explicit negative examples and that it uses intentional background knowledge (Califf and Mooney, 1998). FOIDL has two main learning modes. When the examples are functional, i.e., for each state there is only one best action, FOIDL learns a set of ordered rules, from the more generic to the more specific. When applying the rules only the first most generic rule whose precondition holds needs to be taken into account. When the examples are not functional, i.e., there is at least one state where two actions are equally good, FOIDL learns a bag of rules. All rules whose preconditions hold are applied. Ordered rules are usually easier to understand. In this paper, we use FOIDL in both modes: functional mode for the tutorial example and non-functional mode for the other example.

The rules learned by FOIDL from the optimal strategy are presented in table 2. Preconditions on states express the required and sufficient conditions for the action to be taken for a state. Uppercase letters represent variables (à la Prolog) which can unify with a, b or c. Rules were learned in functional mode. The more generic rules are at the bottom of the table and the more specific at the top. It can be quite clearly seen from these rules that the strategy is composed of two kinds of rules: ordering rules which indicate in what order the variables should be obtained, and generic rules, typeset in italic, which express the strategy of obtaining one piece of information at a time. The ordering, which is a, then b, then c, is arbitrary. It was imposed by the reinforcement learning algorithm. The general strategy consists in preparing to ask whatever piece of information the ordering rules have decided to recognize, and then asking and recognizing a piece of information as soon as we can. By expressing the strategy in the form of rules it is apparent how it operates. It would then be relatively easy for a dialogue engineer to implement a strategy that keeps the optimal one-at-a-time questioning strategy but does not necessarily impose the same order on

Preconditions on state	Action
unknown(a)	prepRec(a)
unknown(b), known(X)	prepRec(b)
known(b), unknown(X)	prepRec(c)
known(c)	end
prepAsk(X)	*ask&recognize*
unknown(X), prepRec(X)	*prepAsk(X)*

Table 2: Tutorial example optimal rules

the process.

3.2 Real world example

Although the tutorial example showed how rules could be obtained and interpreted, it does not say much about the practical use of our approach for real-world dialogues. In order to study this, we applied FOIDL to the optimal strategy presented in (Litman et al., 2000), which "presents a large-scale application of RL [reinforcement learning] to the problem of optimizing dialogue strategy selection [...]". This system is a more realistic application than our introductory example. It has been used by human users over the phone. The dialogue is about activities in New Jersey. A user states his/her preferences for a type of activity (museum visit, etc.) and availability, and the system retrieves potential activities. The system can vary its dialogue strategy by allowing or not allowing users to give extra information when answering a question. It can also decide to confirm or not to confirm a piece of information it has received.

The state of the system represents what pieces of information have been obtained and some information on how the dialogue has evolved so far. This information is represented by variables indicating, in column order in table 3, whether the system has greeted the user (0=no, 1=yes), which piece of information the system wants (1, 2 or 3), what was the confidence in the user's last answer (0=low, 1=medium, 2=high, 3=accept, 4=deny), whether the system got a value for the piece of information (0=no, 1=yes), the number of times the system asked for that piece of information, whether the last question was open-ended (0=open-ended, 1=restrictive), and how well the dialogue was going (0=bad, 1=good).

Preconditions on state							Action	
greetings	info.#	confidence	value	question#	open/closed	history		
B	1	0	1	F	G	H	expconf(1)	
B	1	1	E	F	G	H	expconf(1)	
B	1	2	E	F	G	H	noconf(1)	
B	1	4	E	0	G	H	reaskm(1)	
B	1	D	E	1	G	H	reasks(1)	
B	2	0	1	F	G	H	expconf(2)	
1	2	2	1	0	0	0	noconf(2)	
B	2	0	E	F	1	1	noconf(2)	
B	2	2	E	F	1	H	noconf(2)	
B	2	D	0	1	G	H	reaskm(2)	
B	3	D	1	F	G	1	expconf(3)	
B	3	D	1	F	0	H	expconf(3)	
1	3	1	1	0	1	0	noconf(3)	
1	3	1	1	0	0	1	noconf(3)	
1	3	2	1	0	0	1	noconf(3)	
B	3	0	E	F	G	0	noconf(3)	
B	3	0	E	F	0	H	noconf(3)	
1	*C*	*0*	*0*	*0*	*G*	*H*	*asku(C)*	[2]
B	*C*	*4*	*E*	*0*	*G*	*H*	*reasks(C)*	[3]
B	*C*	*4*	*0*	*F*	*G*	*1*	*reaskm(C)*	[2]
B	*C*	*2*	*E*	*F*	*0*	*1*	*expconf(C)*	[2]
B	*C*	*1*	*1*	*F*	*0*	*H*	*expconf(C)*	[5]
B	*C*	*2*	*E*	*F*	*1*	*0*	*noconf(C)*	[3]
0	*C*	*D*	*E*	*F*	*G*	*H*	*greetu*	[1]

Table 3: NJFun optimal rules

See Litman et al. (2000) for a more detailed explanation of the state representation. The actions the system can take are: greeting users (greetu), asking questions to users (asku), reasking questions to users with an open or restrictive question (reaskm/reasks), asking for confirmation or not (expconf/noconf). The optimal strategy is composed of 42 state-action pairs. It can be reduced to 24 equivalent rules. We present the rules in table 3. Some of these rules are very specific to the state they apply to. The more generic ones, which are valid whatever the exact piece of information being asked, are typeset in italic. The number of states they generalize is indicated in brackets.

These rules can be divided into four categories:

Asking The first rule simply states that asking

(asku) is the best thing to do if we have never asked the value of a piece of information before.

Re-asking The second rule states that the system should re-ask for a value with a restricted grammar (reasks), i.e., a grammar that does not allow mixed-initiative, if the previous attempt was made with an open-ended grammar and the user denied the value obtained. The third rule states that re-asking with an open-ended question (reaskm) is fine when the user denied the value obtained but the dialogue was going well until now.

Confirming The fourth and fifth rules state that the system should explicitly confirm (expconf) a value if the grammar to get it was open-ended, the confidence in the value obtained being medium or even high. No confirmation (noconf) is needed when the confidence is high and the answer was obtained with a restricted grammar even when the dialogue is going badly.

Greeting The last rule indicates that the system should greet the user if it has not done so already.

When preconditions hold for more than one rule, which can for example be the case for reasks and reaskm in some situations, all the actions allowed by the activated rules are possible.

The generic rules are more explicit than the state-based decision table given by reinforcement learning. For example, the rules about asking and greeting are obvious and it is reassuring that the approach suggests them. The effects of open-ended or closed questions on the reasking and confirming policies also become much more apparent. Restricting the potential inputs is the best thing to do when re-asking except if the dialogue was going well until that point. In that case the system can risk having an open-ended grammar. The rules on confirmation show the preference to confirm if the value was obtained via an open-ended grammar and that no confirmation is required if the system has high confidence in a value asked via a closed grammar even if the dialogue is going badly. Because the rules enable us to better understand what the optimal policy does, we may be able to re-use the strategy learned in this specific situation in other dialogue situations.

Iterations	Best score (without rules)	Best score (with rules)
5000	154.13	154.13
10000	153.78	195.97
15000	183.36	220.34
20000	193.47	227.44 (*)
25000	210.77	231.43 (*)
30000	224.32	231.68 (*)
35000	224.40	231.71 (*)
40000	228.60	231.72 (*)
45000	228.95	231.72 (*)
50000	230.44 (*)	231.72 (*)

Table 4: Effect of using rules during learning

It should be noted that the generic rules generalize only a part of the total strategy (18 states out of 42 in the example). Therefore a lot remains to be explained about the less generic rules. For example, the second piece of information does not require confirmation even if we got it with a low confidence value if the grammar was restrictive and the dialogue going well. Under the same conditions the first piece of information would require a confirmation. The underlying reasons for these differences are not clear. Some of the decisions made by the reinforcement learning algorithm are also hard to explain, whether in the form of rules or not. For example, the optimal strategy states that the third piece of information does not require confirmation if we got it with low confidence and the dialogue was going badly. It is difficult to explain why this is the best action to take.

4 Learning optimal strategy using rules

In this section, we discuss the use of rules during learning. Since rules can generalize the optimal strategy as we saw in the previous section, it is interesting to see whether they can generalize strategies obtained during training. If the rules can generalize the up-to-now best strategy, we may then be able to benefit from the rules to guide the search for the optimal strategy throughout the search space. In order to test this, we ran the same reinforcement learning algorithm to find out the optimal policy in

the same setting as in the example system of section 3. We also ran the same algorithm but this time we stopped it every 5000 iterations. An iteration corresponds to a transition between states in the dialogue. We then searched for rules summarizing the best policy found until then. We took the generic rules found, i.e., not the ones that are specific to a particular state, and used these to direct the search. That is to say, when a rule applied we chose to take the action it suggested rather than the action suggested by the state's values (this is still subjected to the 0.8 probability selection). The idea behind this was that, if the rules generalize correctly the best strategy, following the rules would guide us more quickly to the best policy than a blind exploration. It should be noted that the underlying representation is still state-based, i.e., we do not generalize the state evaluation function. Our method is therefore guaranteed to find the optimal policy even if the actions suggested by the rules are not the right ones.

Table 4 summarizes the value of the best policy found after each step of 5000 iterations. A star (*) indicates that the optimal strategy has been consistently found. As can be seen from this table, using rules during learning improved the value of the best strategy found so far and reduced the number of iterations needed to find the optimal strategy for this particular example. The main effect of using rules seems to be the stabilization of the search on the optimal policy. The search without rules finds the optimal policy but then goes off track before coming back to it. This may not be always the case[1] since the best strategy found at first may not be optimal at all (for example, a rather good strategy at first is to end the dialogue immediately since it avoids negative rewards), or the dialogue may not be regular enough for rules to be useful. In these cases using rules may well be detrimental. Nevertheless it is important to see that rules can help reduce, in this case by a factor of 2, the number of iterations needed to find the optimal strategy. Computationally, using rules may not be much different than not using them since the benefits of fewer reinforcement learning cycles are counter-balanced by the inductive learning costs. However, requiring fewer training dialogues is still an important advantage of this method. This is especially true for systems that train online with real users rather than simulated ones. In this case, example dialogues are an expensive commodity and reducing the need for training dialogues is beneficial.

[1]We do not claim any statistical evidence since we ran only a limited set of experiments on the effects of rules and present just one here. Even if we ran enough experiments to get statistically significant results, they would be of little use as they would depend on a particular type of dialogues. Much more work needs to be done to evaluate the influence of rules on reinforcement learning and, if possible, in which conditions they are useful.

5 Discussion

Recent work on reinforcement learning and dialogue management has mainly focused on how to reduce the search space for the optimal strategy. Because reinforcement learning is state based and there may potentially be a large number of states, problems may arise when few dialogues are available and the data too sparse to select the best strategy. States can usually be collapsed to make this problem less acute. The main idea here is to express the state of the dialogue by a limited number of features while keeping enough and the right kind of information to be able to learn useful strategies (Walker et al., 1998). There has also been new research on how to model the dialogue with partially observable Markov models (Roy et al., 2000).

Some work has also been done on finding out rules to select dialogue management strategies. For example, Litman and Pan (2000) use machine learning to learn rules detecting when dialogues go badly. The dialogue manager uses a strategy predefined by a dialogue designer. If a rule detects a bad dialogue, the dialogue strategy is changed to a more restrictive, more system guided strategy. Our approach is different from that work since the strategy is not predefined but based on the optimal strategy found by reinforcement learning. Our rules not only detect in principle when a dialogue is going badly but also indicate which action to take. The efficiency of the rules obviously depends on the way the optimal strategy search space has been modeled and other conditions influencing learning.

Some pieces of work have been concerned with natural language processing from an induc-

tive logic programming point of view. Notably, work on morphology (Mooney and Califf, 1995) and parsing (Thompson et al., 1997) has been carried out. However, as far as we know, the application of inductive logic programming to dialogue management is new.

6 Conclusion

In this paper, we presented an approach for finding and expressing optimal dialogue strategies. We suggested using inductive logic programming to generalize the results given by reinforcement learning methods. The resulting rules are more explicit than the decision tables given by reinforcement learning alone. This allows dialogue designers to better understand the effect of the optimal strategy and improves potential re-use of the strategies learned. We also show that in some situations rules may have a beneficial effect when used during learning. By guiding the search based on the best strategy found so far, they can direct a reinforcement learning program towards the optimal strategy, thus reducing the amount of training dialogues needed. More work needs to be done to determine, if possible, under which conditions such improvements can be obtained.

References

Niels Ole Bernsen, Hans Dybkjær, and Laila Dybkjær. 1998. *Designing interactive speech systems*. Springer-Verlag.

Mary Elaine Califf and Raymond J. Mooney. 1998. Advantages of decision lists and implicit negatives in inductive logic programming. *New Generation Computing*, 16(3):263–281.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. Technical Report 97.28.1, AT&T Labs Research.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1998. Using Markov decision process for learning dialogue strategies. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing, Seattle, USA*, May.

Diane J. Litman and Shimei Pan. 2000. Predicting and adapting to poor speech recognition in a spoken dialogue system. In *Proceedings of the* 17^{th} *national conference on artificial intelligence, Austin, Texas, USA*, August.

Diane J. Litman, Michael S. Kearns, Satinder B. Singh, and Marilyn A. Walker. 2000. Automatic optimization of dialogue management. In *Proceedings of the* 18^{th} *international conference on computational linguistics, Saarbrücken, Luxembourg, Nancy*, July.

Tom M. Mitchell. 1997. *Machine learning*. McGraw-Hill.

Raymond J. Mooney and Mary Elaine Califf. 1995. Induction of first-order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, 3:1–24.

Nicholas Roy, Joelle Pineau, and Sebastian Thrun. 2000. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the* 38^{th} *annual meeting of the Association for Computational Linguistics, Hong-Kong*, October.

Satinder B. Singh, Michael S. Kearns, Diane J. Litman, and Marylin A. Walker. 2000. Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proceedings of the* 17^{th} *national conference on artificial intelligence, Austin, USA*, July.

Cynthia A. Thompson, Raymond J. Mooney, and Lappoon R. Tang. 1997. Learning to parse natural language database queries into logical forms. In *Proceedings of the workshop on automata induction, grammatical inference, and language acquisition, Nashville, Tennessee, USA*, July.

Marilyn A. Walker, Jeanne C. Fromer, and Shrikanth Narayanan. 1998. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *Proceedings of the* 17^{th} *international conference on computational linguistics and the* 36^{th} *annual meeting of the Association for Computational Linguistics, Montreal, Quebec, Canada*, August.

Marilyn Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416.