Evolving optimal inspectable strategies for spoken dialogue systems

Dave Toney
School of Informatics
Edinburgh University
2 Buccleuch Place
Edinburgh EH8 9LW
dave@cstr.ed.ac.uk

Johanna Moore
School of Informatics
Edinburgh University
2 Buccleuch Place
Edinburgh EH8 9LW
jmoore@inf.ed.ac.uk

Oliver Lemon
School of Informatics
Edinburgh University
2 Buccleuch Place
Edinburgh EH8 9LW
olemon@inf.ed.ac.uk

Abstract

We report on a novel approach to generating strategies for spoken dialogue systems. We present a series of experiments that illustrate how an *evolutionary* reinforcement learning algorithm can produce strategies that are both optimal and easily inspectable by human developers. Our experimental strategies achieve a mean performance of 98.9% with respect to a predefined evaluation metric. Our approach also produces a dramatic reduction in strategy size when compared with conventional reinforcement learning techniques (87% in one experiment). We conclude that this algorithm can be used to evolve optimal inspectable dialogue strategies.

1 Introduction

Developing a dialogue management strategy for a spoken dialogue system is often a complex and time-consuming task. This is because the number of unique conversations that can occur between a user and the system is almost unlimited. Consequently, a system developer may spend a lot of time anticipating how potential users might interact with the system before deciding on the appropriate system response.

Recent research has focused on generating dialogue strategies automatically. This work is based on modelling dialogue as a markov decision process, formalised by a finite state space *S*, a finite action set *A*, a set of transition probabilities *T* and a reward function *R*. Using this model an optimal dialogue strategy π^* is represented by a mapping between the state space and the action set. That is, for each state $s \in S$ this mapping defines its optimal action a_s^*. How is this mapping constructed? Previous approaches have employed reinforcement learning (RL) algorithms to estimate an optimal value function Q^* (Levin et al., 2000; Frampton and Lemon, 2005). For each state this function predicts the future reward associated with each action available in that state. This function makes it easy to extract the optimal strategy (*policy* in the RL literature).

Progress has been made with this approach but some important challenges remain. For instance, very little success has been achieved with the large state spaces that are typical of real-life systems. Similarly, work on summarising learned strategies for interpretation by human developers has so far only been applied to tasks where each state-action pair is explicitly represented (Lecœuche, 2001). This tabular representation severely limits the size of the state space.

We propose an alternative approach to finding optimal dialogue policies. We make use of XCS, an *evolutionary* reinforcement learning algorithm that seeks to represent a policy as a compact set of state-action rules (Wilson, 1995). We suggest that this algorithm could overcome both the challenge of large state spaces and the desire for strategy inspectability. In this paper, we focus on the issue of inspectability. We present a series of experiments that illustrate how XCS can be used to evolve dialogue strategies that are both optimal and easily inspectable.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 173–176,
New York, June 2006.

2 Learning Classifier Systems and XCS

Learning Classifier Systems were introduced by John Holland in the 1970s as a framework for learning rule-based knowledge representations (Holland, 1976). In this model, a rule base consists of a population of *N* state-action rules known as *classifiers*. The state part of a classifier is represented by a ternary string from the set {0,1,#} while the action part is composed from {0,1}. The # symbol acts as a wildcard allowing a classifier to aggregate states; for example, the state string 1#1 matches the states 111 and 101. Classifier systems have been applied to a number of learning tasks, including data mining, optimisation and control (Bull, 2004).

Classifier systems combine two machine learning techniques to find the optimal rule set. A genetic algorithm is used to evaluate and modify the population of rules while reinforcement learning is used to assign rewards to existing rules. The search for better rules is guided by the *strength* parameter associated with each classifier. This parameter serves as a fitness score for the genetic algorithm and as a predictor of future reward (*payoff*) for the RL algorithm. This evolutionary learning process searches the space of possible rule sets to find an optimal policy as defined by the reward function.

XCS (X Classifier System) incorporates a number of modifications to Holland's original framework (Wilson, 1995). In this system, a classifier's fitness is based on the accuracy of its payoff prediction instead of the prediction itself. Furthermore, the genetic algorithm operates on actions instead of the population as a whole. These aspects of XCS result in a more complete map of the state-action space than would be the case with strength-based classifier systems. Consequently, XCS often outperforms strength-based systems in sequential decision problems (Kovacs, 2000).

3 Experimental Methodology

In this section we present a simple slot-filling system based on the hotel booking domain. The goal of the system is to acquire the values for three slots: the check-in date, the number of nights the user wishes to stay and the type of room required (single, twin etc.). In slot-filling dialogues, an optimal strategy is one that interacts with the user in a satisfactory way while trying to minimise the length of the dialogue. A fundamental component of user satisfaction is the system's prevention and repair of any miscommunication between it and the user. Consequently, our hotel booking system focuses on evolving essential slot confirmation strategies.

We devised an experimental framework for modelling the hotel system as a sequential decision task and used XCS to evolve three behaviours. Firstly, the system should execute its dialogue acts in a logical sequence. In other words, the system should greet the user, ask for the slot information, present the query results and then finish the dialogue, in that order (Experiment 1). Secondly, the system should try to acquire the slot values as quickly as possible while taking account of the possibility of misrecognition (Experiments 2a and 2b). Thirdly, to increase the likelihood of acquiring the slot values correctly, each one should be confirmed at least once (Experiments 3 and 4).

The reward function for Experiments 1, 2a and 2b was the same. During a dialogue, each non-terminal system action received a reward value of zero. At the end of each dialogue, the final reward comprised three parts: (i) -1000 for each system turn; (ii) 100,000 if all slots were filled; (iii) 100,000 if the first system act was a greeting. In Experiments 3 and 4, an additional reward of 100,000 was assigned if all slots were confirmed.

The transition probabilities were modelled using two versions of a handcoded simulated user. A very large number of test dialogues are usually required for learning optimal dialogue strategies; simulated users are a practical alternative to employing human test users (Scheffler and Young, 2000; Lopez-Cozar et al., 2002). Simulated user A represented a fully cooperative user, always giving the slot information that was asked. User B was less cooperative, giving no response 20% of the time. This allowed us to perform a two-fold cross validation of the evolved strategies.

For each experiment we allowed the system's strategy to evolve over 100,000 dialogues with each simulated user. Dialogues were limited to a maximum of 30 system turns. We then tested each strategy with a further 10,000 dialogues. We logged the total reward (payoff) for each test dialogue. Each experiment was repeated ten times.

In each experiment, the presentation of the query results and closure of the dialogue were combined into a single dialogue act. Therefore, the dialogue acts available to the system for the first experiment were: *Greeting*, *Query+Goodbye*, *Ask(Date)*, *Ask(Duration)* and *Ask(RoomType)*. Four boolean variables were used to represent the state of the dialogue: *GreetingFirst*, *DateFilled*, *DurationFilled*, *RoomFilled*.

Experiment 2 added a new dialogue act: *Ask(All)*. The goal here was to ask for all three slot values if the probability of getting the slot values was reasonably high. If the probability was low, the system should ask for the slots one at a time as before. This information was modelled in the simulated users by 2 variables: *Prob1SlotCorrect* and *Prob3SlotsCorrect*. The values for these variables in Experiments 2a and 2b respectively were: 0.9 and 0.729 ($=0.9^3$); 0.5 and 0.125 ($=0.5^3$).

Experiment 3 added three new dialogue acts: *Explicit_Confirm(Date)*, *Explicit_Confirm(Duration)*, *Explicit_Confirm(RoomType)* and three new state variables: *DateConfirmed*, *DurationConfirmed*, *RoomConfirmed*. The goal here was for the system to learn to confirm each of the slot values after the user has first given them. Experiment 4 sought to reduce the dialogue length further by allowing the system to confirm one slot value while asking for another. Two new dialogue acts were available in this last experiment: *Implicit_Confirm(Date)+Ask(Duration)* and *Implicit_Confirm(Duration)+Ask(RoomType)*.

4 Experimental Results

Table 1 lists the total reward (payoff) averaged over the 10 cross-validated test trials for each experiment, expressed as a percentage of the maximum payoff. In these experiments, the maximum payoff represents the shortest possible successful dialogue. For example, the maximum payoff for Experiment 1 is 195,000: 100,000 for filling the slots plus 100,000 for greeting the user at the start of the dialogue minus 5000 for the minimum number of turns (five) taken to complete the dialogue successfully. The average payoff for the 10 trials trained on simulated user A and tested on user B was 193,877 – approximately 99.4% of the maximum possible. In light of these results and the stochastic user responses, we suggest that these evolved strategies would compare favourably with any handcoded strategies.

Exp.	Training/Test Users	Payoff (%)
1	A, B	99.4
	B, A	99.8
2a	A, B	99.1
	B, A	99.4
2b	A, B	96.8
	B, A	97.2
3	A, B	98.8
	B, A	99.3
4	A, B	99.3
	B, A	99.7

Table 1: Payoff results for the evolved strategies.

It is instructive to compare the rate of convergence for different strategies. Figure 1 shows the average payoff for the 100,000 dialogues trained with simulated user A in Experiments 3 and 4. It shows that Experiment 3 approached the optimal policy after approximately 20,000 dialogues whereas Experiment 4 converged after approximately 5000 dialogues. This is encouraging because it suggests that XCS remains focused on finding the shortest successful dialogue even when the number of available actions increases.

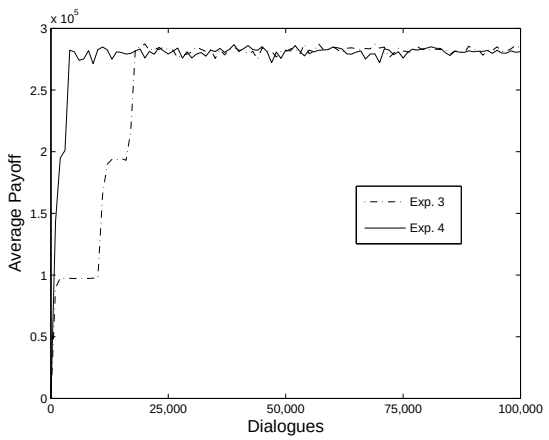

Figure 1: Convergence towards optimality during training in Experiments 3 and 4 (simulated user A).

Finally, we look at how to represent an optimal strategy. From the logs of the test dialogues we extracted the state-action rules (classifiers) that were executed. For example, in Experiment 4, the op-

State							Action
GreetingFirst	*DateFilled*	*DurationFilled*	*RoomFilled*	*DateConfirmed*	*DurationConfirmed*	*RoomConfirmed*	
0	0	#	#	#	#	#	*Greeting*
1	0	0	0	#	#	#	*Ask(Date)*
1	1	#	#	0	#	#	*Implicit_Confirm(Date) + Ask(Duration)*
1	1	1	#	1	0	0	*Implicit_Confirm(Duration) + Ask(RoomType)*
1	1	1	1	1	1	0	*Explicit_Confirm(RoomType)*
1	1	1	1	1	1	1	*Query + Goodbye*

Table 2: A summary of the optimal strategy for Experiment 4.

timal strategy is represented by 17 classifiers. By comparison, a purely RL-based strategy would define an optimal action for every theoretically possible state (i.e. 128). In this example, the evolutionary approach has reduced the number of rules from 128 to 17 (a reduction of 87%) and is therefore much more easily inspectable. In fact, the size of the optimal strategy can be reduced further by selecting the most general classifier for each action (Table 2). These rules are sufficient since they cover the 60 states that could actually occur while following the optimal strategy.

5 Conclusions and future work

We have presented a novel approach to generating spoken dialogue strategies that are both optimal and easily inspectable. The generalizing ability of the evolutionary reinforcement learning (RL) algorithm, XCS, can dramatically reduce the size of the optimal strategy when compared with conventional RL techniques. In future work, we intend to exploit this generalization feature further by developing systems that require much larger state representations. We also plan to investigate other approaches to strategy summarisation. Finally, we will evaluate our approach against purely RL-based methods.

References

Larry Bull, editor. 2004. *Applications of Learning Classifier Systems*. Springer.

Matthew Frampton and Oliver Lemon. 2005. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Edinburgh, UK, July.

John Holland. 1976. Adaptation. In Rosen R. and F. Snell, editors, *Progress in theoretical biology*. Plenum, New York.

Tim Kovacs. 2000. Strength or accuracy? Fitness calculation in learning classifier systems. In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart Wilson, editors, *Learning Classifier Systems. From Foundations to Applications*, Lecture Notes in Artificial Intelligence 1813, pages 143–160. Springer-Verlag.

Renaud Lecœuche. 2001. Learning optimal dialogue management rules by using reinforcement learning and inductive logic programming. In *2nd Meeting of the North American Chapter of the Association of Computational Linguistics*, Pittsburgh, USA, June.

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

R. Lopez-Cozar, A. De la Torre, J. Segura, A. Rubio, and V. Sánchez. 2002. Testing dialogue systems by means of automatic generation of conversations. *Interacting with Computers*, 14(5):521–546.

Konrad Scheffler and Steve Young. 2000. Probabilistic simulation of human-machine dialogues. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1217–1220, Istanbul, Turkey, June.

Stewart Wilson. 1995. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.