Developing Conversational Interfaces with XCS

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

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

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

ABSTRACT

Conversational interfaces allow human users to interact with computer systems using spoken language in order to retrieve information and perform problem-solving tasks. A crucial part of developing such a system is devising a conversational strategy. This strategy is effectively a mapping between anticipated user inputs and appropriate system outputs. In this paper we present some preliminary experiments that show how XCS can generate optimal conversational strategies. We also explore the effect of the magnitude and structure of reward functions on the strategies learned by XCS.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning–Concept learning; I.2.7 [**Artificial Intelligence**]: Natural Language Processing–Discourse.

General Terms

Algorithms, Design, Experimentation, Performance.

Keywords

Learning Classifier Systems, XCS, Conversational Interfaces, Spoken Dialogue Systems, Conversational Strategies.

1. INTRODUCTION

Since the early 1990s, speech- and language-processing technologies have been combined to create conversational interfaces (CIs). These interfaces allow human users to converse with a computer to retrieve information and perform problem-solving tasks. However, developing a conversational interface can be a difficult and time-consuming process. Although each CI is normally developed for a specific purpose (e.g. to provide a weather report) 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 most appropriate system responses. These decisions are encoded in a *conversational strategy*, effectively a mapping between anticipated user inputs and appropriate system outputs.

To reduce the time and effort associated with developing a conversational strategy, recent work has concentrated on modelling dialogue as a sequential decision problem. Using this model, reinforcement learning algorithms have been employed to generate conversational strategies automatically. Some progress has been made with this approach. However, a number of important challenges remain.

In this paper we describe how we have begun to use XCS [14] to generate conversational strategies. We outline some of the challenges associated with learning strategies for CIs. We believe that XCS might allow us to overcome many of these issues. We present some preliminary experiments that show how XCS can generate strategies that are optimal with respect to pre-defined evaluation criteria. We also present results that illustrate the problem of defining appropriate reward functions for conversational systems.

2. CONVERSATIONAL INTERFACES

Conversational interfaces (also known as spoken dialogue systems) are based on the idea that spoken language can serve as a flexible and efficient form of communication between humans and computers. Information retrieval and transactional systems are arguably the most prevalent forms of CI in commercial use today. Interaction is almost exclusively telephony-based. Existing applications allow users to retrieve tourist, weather and telephone directory information, make travel reservations and conduct financial transactions [9]. A simple example of a typical interaction between a user and a train information system is shown in Figure 1.

System:	Welcome to National Rail Enquiries. How can I help you?
User:	I'd like to travel to Baltimore tomorrow.
System:	Baltimore, leaving from which station?
User:	Washington, around seven am.
System:	There is a train from Washington Union Station to Baltimore Penn Station at 7:00 am. It arrives at 7:35. Is this ok?
User:	Yes, thank you. Goodbye.
System:	Goodbye.

Figure 1: A simple train information dialogue.

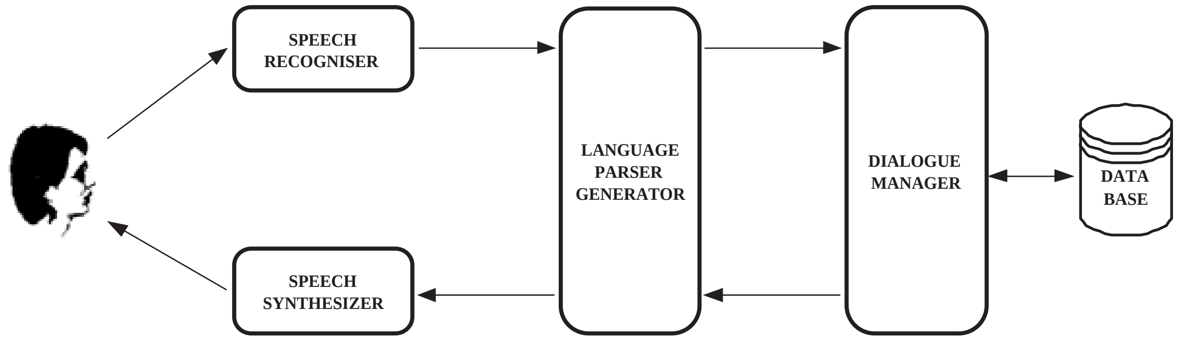

Figure 2: Core technologies in a conversational interface.

The example in Figure 1 is basically a database query: the user provides the values for some database fields (*slots*) and the system returns the query results. Consequently, this type of system is often referred to as a *slot-filling* system. From the user's perspective the interaction is conducted with a single computer system. In reality, a CI is comprised of a number of core technologies. Typically, a CI will include a speech recogniser, a speech synthesizer, a language parser/generator and a dialogue manager (see Figure 2).

A single exchange between user and system normally involves the following stages: (i) the user's utterance is sampled and processed by the speech recogniser; the recogniser generates a list of potential sentences and associated confidence scores; (ii) this list is processed by the language parser and encoded into a semantic form that can be interpreted by the dialogue manager (DM); (iii) the DM selects the most likely interpretation of the user's utterance within the context of the dialogue as a whole and then generates its own response; (iv) the DM's response is translated by the language generator into text which in turn is converted to a speech waveform by the speech synthesizer. The DM may occasionally consult with a backend database when forming its response.

How does the DM interpret the user's utterances and select appropriate responses? These decisions are encoded in the DM's conversational strategy. This strategy is devised by the system developer and represents a policy for responding to expected user input. In other words, the DM executes the developer's conversational strategy. Many of the CI's subsystems require little change when moving from one application to another. However, the conversational strategy is primarily what differentiates one system from another. Consequently, a large proportion of a system's development is taken up with strategy design.

3. CONVERSATIONAL STRATEGIES

The traditional approach to CI development is to evaluate successive versions of the conversational strategy with test users until a required level of performance is achieved. For a particular system, this test-and-refine process may be repeated many times. Furthermore, the number of strategies that has been explored may be considerably less than the total number of possible strategies. To overcome these two issues, recent research has focused on generating dialogue strategies automatically. This work is based on modelling dialogue as a finite Markov Decision Process, formalized by a state set S, an action set A, a set of transition probabilities T and a reward function R . Using this model, conversational strategies have been developed using reinforcement learning (RL) algorithms [7, 12].

In the context of CIs, the dialogue state represents the progress of the dialogue, for example, how many slot values have been supplied by the user, boolean variables indicating whether the user has been greeted by the system etc. Therefore, the state set simply enumerates all possible dialogue states. The action set is the list of possible system actions (utterances), such as asking the user for information ("What is your destination?"). Transition probabilities, which describe the effect of user actions on the dialogue state, are often modelled by a *simulated user*. A very large number of test dialogues are usually required by learning algorithms to generate conversational strategies; simulated users are a practical alternative to employing human test users [8, 10]. Finally, the reward function allows the system developer to define what characterizes a successful dialogue.

Some progress has been made with the use of RL techniques to generate conversational strategies. However, a number of important challenges remain. For instance, very little success has been achieved with the large state sets that are typical of real-life systems (but see [4]). Similarly, work on summarizing learned strategies for interpretation by human developers has so far only been applied to tasks where each state-action pair is explicitly represented [6]. This tabular representation severely limits the size of the state space. In addition, misrecognition on the part of the speech recogniser can create situations where the DM makes decisions based on an inaccurate assessment of the dialogue state. In other words, the true state of the dialogue is only partially observable by the DM. Attempts to incorporate a model of this partial observability into the conversational strategy have so far succeeded with only very small state representations [13]. We believe that XCS and its variants (e.g.[5, 15]) may provide solutions to many of these challenges.

4. EXPERIMENTAL METHODOLOGY

In this section we present a simple slot-filling system based on train information enquiries. The goal of the system is to acquire the values for four slots: the departure city, the destination city, the date of departure and the preferred time of departure. In slot-filling dialogues, an optimal strategy is one that interacts with the user in a sensible way while trying to minimise the length of the dialogue. A common example of sensible system behaviour is the practice of confirming slot values when speech recognition confidence is generally low, for example, when the level of ambient noise is high. In our train information system we have assumed that such a situation exists. Therefore, the optimal strategy consists of asking the user for values for each of the four slots, confirming each slot value, presenting the query results and then closing the dialogue.

We devised an experimental framework for modelling the train information system as a Markov Decision Process and used XCS to generate conversational strategies. Our state representation consisted of eight boolean flags indicating whether each of the slots have been filled and confirmed: *Departure_Filled, Destination_Filled, Date_Filled, Time_Filled, Departure_Confirmed, Destination_Confirmed, Date_Confirmed, Time_Confirmed*. The available system actions were: *Ask(Departure_City), Ask(Destination_City), Ask(Date), Ask(Preferred_Time), Confirm(Departure_City), Confirm(Destination_City), Confirm(Date), Confirm(Preferred_Time)* and *Query+Goodbye*. The transitions probabilities were implemented using a handcoded simulated user.

In this work we were particularly interested in the effect of the reward function on the policy generated by XCS. In conversational systems, reward functions are normally designed to balance the need for sensible system behaviour and the desire for short dialogues. With respect to this goal, previous work has investigated the relative size in magnitude between interim and final rewards using Q-learning [2]. We investigated two further aspects of the reward function. In the first set of experiments, we examined the effect of varying the *absolute* magnitude of the rewards. We created six versions of the reward function. Each function maintained the same ratio between interim and final rewards but differed in absolute terms. At the end of each dialogue a penalty was assigned to each action performed and a larger reward was given if the goal of filling and confirming the four slots was achieved (Table 1). For example, the total possible payoff associated with the optimal policy in Experiment 1a was: $-1 \times 9 + 100 = 91$, i.e. -1 for each of the nine actions required to complete the dialogue and 100 for filling and confirming the four slots.

Exp.	Per Action	Goal Completion	Max Payoff
1a	-1	100	91
1b	-10	1,000	910
1c	-100	10,000	9,100
1d	-1000	100,000	91,000
1e	-10,000	1,000,000	910,000
1f	-100,000	10,000,000	9,100,000

Table 1: Reward functions for Experiment Set 1.

In the second set of experiments, we examined the effect of partial rewards. We wanted to know if dividing the reward into sub-rewards affected the rate of convergence towards the optimal policy. Previous work has shown that applying partial rewards to tabular Q-learning can produce suboptimal conversational strategies [3]. Experiment 2a had the same reward structure as Experiment 1c, with a total possible payoff of: $-100 \times 9 + 10,000 = 9,100$. In Experiments 2b and 2c, the total possible payoff was also 9,100 (Table 2). However, the reward functions were structured differently. In Experiment 2b a reward of 5,000 was given if all four slots were filled; a further 5,000 was awarded if all four slots values were confirmed. In Experiment 2c, a reward of 2,500 was awarded for filling and confirming each slot. Therefore, the total payoff for filling and confirming all four slots in Experiments 2b and 2c was identical to 2a.

Exp.	Per Action	Goal Completion	Max Payoff
2a	-100	fill and confirm all slots:10,000	9,100
2b	-100	fill all slots:5,000 confirm all slots:5,000	9,100
2c	-100	fill and confirm one slot:2,500	9,100

Table 2: Reward functions for Experiment Set 2.

In each experiment, dialogues were limited to a maximum of 30 system actions. Each experiment was terminated after 25,000 training dialogues and repeated ten times. The XCS parameter settings (described in [1]) in all experiments were: $N = 1000$, $\beta = 0.2$, $\alpha = 0.1$, $\epsilon_0 = 10$, $\nu = 5$, $\gamma = 0.95$, $\theta_{GA} = 25$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 20$, $P_{\#} = 0.33$, $p_I = 12$, $\epsilon_I = 0$, $F_I = 0.01$, $p_{explr} = 0.5$, $\theta_{mna} = 9$.

5. EXPERIMENTAL RESULTS

Figure 3 shows the average payoff over 25,000 (exploit) training dialogues for Experiments 1a–1f. Note that an optimal policy was not achieved in 1a and 1b. In fact, both these experiments were allowed to run for 10^7 dialogues without improvement. The average payoff of 100 in 1b only occurs because of the explore mechanism within XCS and is well short of the optimal payoff of 910. However, in Experiments 1c–1f, the optimal policy was reached fairly rapidly, after approximately 2,000–9,000 dialogues. Increasing the magnitude of the rewards beyond -100/10,000 (used in 1c) seems to have little effect on the rate of convergence.

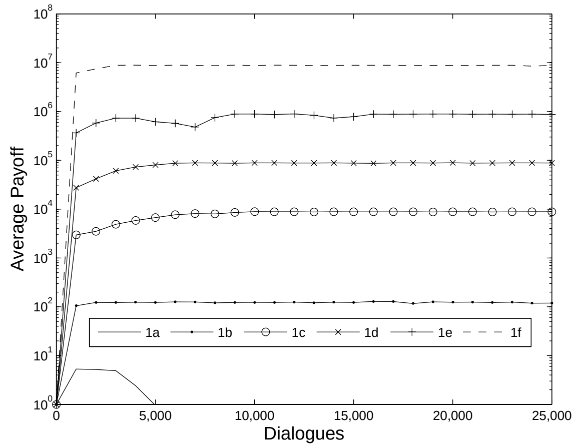

Figure 3: Policy convergence: Experiments 1a–1f.

Figure 4 shows the average payoff over 25,000 (exploit) training dialogues for Experiments 2a–2c. Recall that the full reward of 10,000 was given in Experiment 2a *only* when all slots have been filled and confirmed. The optimal policy was reached after approximately 9,000 dialogues. However, in Experiment 2b, the average payoff converged to just over 6,000. This reflects the fact that the algorithm was oscillating between a payoff of 4,500 (indicating that all slots have been filled) and 9,100 (indicating that all slots have also been confirmed). Therefore, the algorithm has not settled on the optimal policy. In Experiment 2c, where a partial reward of 2,500 was awarded to each slot that is filled and confirmed, the optimal policy *was* reached. In fact, the policy converged to optimality more quickly than in Experiment 2a, after approximately 2,000 dialogues.

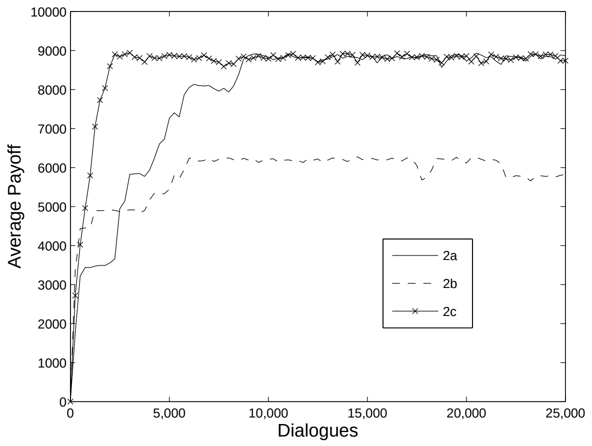

Figure 4: Policy convergence: Experiments 2a–2c.

6. CONCLUSIONS AND FUTURE WORK

We have shown that XCS can be used to generate optimal strategies for conversational interfaces. However, for this approach to be useful, the ability to generate optimal strategies must be predictable and consistent. Our preliminary results show that a policy generated by XCS can be affected by both the magnitude and structure of the reward function. It may be that small rewards take too long to increase the fitness of useful macroclassifiers. In terms of subdividing the reward function, our initial results have not shown a clear pattern. One tentative suggestion is that these effects may be caused by the relationship between the positional semantics of the state representation and the partial reward structure (cf.[11]). We plan to investigate this idea further. Notwithstanding this, similar problems do occur when using other algorithms, such as Q-learning, to learn conversational strategies [3]. Designing reliable reward functions for generating conversational strategies is not easy.

For the future, a great deal of work remains to be done. A robust model of reward is crucial if a software tool is to be developed for reliably generating conversational strategies. We also need to develop strategies that require much larger state and action sets. We plan to investigate the efficacy of XCS variants (e.g. XCSμ [5]) that tackle the issue of partial observability. Finally, it would be very useful to be able to summarise optimal strategies for inspection by human developers. We may be able to adapt a variety of approaches (e.g. [6, 15]) for this purpose.

7. REFERENCES

[1] M. Butz and S. Wilson. An algorithmic description of XCS. *Soft Computing*, 6:144–153, 2002.

[2] M. English and P. Heeman. Learning mixed initiative dialog strategies by using reinforcement learning on both conversants. In *Conference on Empirical Methods in Natural Language Processing*, Vancouver, Canada, October 2005.

[3] M. Frampton and O. Lemon. 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 2005.

[4] J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data. In *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Edinburgh, UK, July 2005.

[5] P. L. Lanzi. An extension to the XCS classifier system for stochastic environments. In *Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 353–360, Orlando, FL, USA, July 1999.

[6] R. Lecœuche. 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 2001.

[7] E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialogue strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.

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

[9] Nuance. http://www.nuance.com.

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

[11] D. Schuurmans and J. Schaeffer. Representational difficulties with classifier systems. In *3rd International Conference on Genetic Algorithms*, pages 328–333, Fairfax, VA, USA, June 1989.

[12] S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

[13] J. Williams, P. Poupart, and S. Young. Partially obervable markov decision processes with continuous observations for dialogue management. In *6th SigDial Workshop on Discourse and Dialogue*, Lisbon, Portugal, September 2005.

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

[15] S. Wilson. Compact rulesets from XCSI. In *4th International Workshop on Advances in Learning Classifier Systems*, pages 197–210, San Francisco, CA, USA, July 2001.