Learning More Effective Dialogue Strategies Using Limited Dialogue Move Features

Matthew Frampton and **Oliver Lemon**
HCRC, School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, UK
M.J.E.Frampton@sms.ed.ac.uk, olemon@inf.ed.ac.uk

Abstract

We explore the use of restricted dialogue contexts in reinforcement learning (RL) of effective dialogue strategies for information seeking spoken dialogue systems (e.g. COMMUNICATOR (Walker et al., 2001)). The contexts we use are richer than previous research in this area, e.g. (Levin and Pieraccini, 1997; Scheffler and Young, 2001; Singh et al., 2002; Pietquin, 2004), which use only slot-based information, but are much less complex than the full dialogue "Information States" explored in (Henderson et al., 2005), for which tractabe learning is an issue. We explore how incrementally adding richer features allows learning of more effective dialogue strategies. We use 2 user simulations learned from COMMUNICATOR data (Walker et al., 2001; Georgila et al., 2005b) to explore the effects of different features on learned dialogue strategies. Our results show that adding the dialogue moves of the last system and user turns increases the average reward of the automatically learned strategies by 65.9% over the original (hand-coded) COMMUNICATOR systems, and by 7.8% over a baseline RL policy that uses only slot-status features. We show that the learned strategies exhibit an emergent "focus switching" strategy and effective use of the 'give help' action.

1 Introduction

Reinforcement Learning (RL) applied to the problem of dialogue management attempts to find optimal mappings from dialogue contexts to system actions. The idea of using *Markov Decision Processes (MDPs)* and reinforcement learning to design dialogue strategies for dialogue systems was first proposed by (Levin and Pieraccini, 1997). There, and in subsequent work such as (Singh et al., 2002; Pietquin, 2004; Scheffler and Young, 2001), only very limited state information was used in strategy learning, based always on the number and status of filled information slots in the application (e.g. departure-city is filled, destination-city is unfilled). This we refer to as *low-level* contextual information. Much prior work (Singh et al., 2002) concentrated only on specific strategy decisions (e.g. confirmation and initiative strategies), rather than the full problem of what system dialogue move to take next.

The simple strategies learned for low-level definitions of state cannot be sensitive to (sometimes critical) aspects of the dialogue context, such as the user's last dialogue move (DM) (e.g. request-help) unless that move directly affects the status of an information slot (e.g. *provide-info(destination-city)*). We refer to additional contextual information such as the system and user's last dialogue moves as *high-level* contextual information. (Frampton and Lemon, 2005) learned full strategies with limited 'high-level' information (i.e. the dialogue move(s) of the last user utterance) and only used a stochastic user simulation whose probabilities were supplied via commonsense and intuition, rather than learned from data. This paper uses data-driven n-gram user simulations (Georgila et al., 2005a) and a richer dialogue context.

On the other hand, increasing the size of the state space for RL has the danger of making the learning problem intractable, and at the very least means that data is more sparse and state approximation methods may need to be used (Henderson et al., 2005). To date, the use of very large state spaces relies on a "hybrid" supervised/reinforcement learning technique, where the reinforcement learning element has not yet been shown to significantly improve policies over the purely supervised case (Henderson et al., 2005).

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 185–192,
Sydney, July 2006.

The extended state spaces that we propose are based on theories of dialogue such as (Clark, 1996; Searle, 1969; Austin, 1962; Larsson and Traum, 2000), where which actions a dialogue participant can or should take next are not based solely on the task-state (i.e. in our domain, which slots are filled), but also on wider contextual factors such as a user's dialogue moves or speech acts. In future work we also intend to use feature selection techniques (e.g. correlation-based feature subset (CFS) evaluation (Rieser and Lemon, 2006)) on the COMMUNICATOR data (Georgila et al., 2005a; Walker et al., 2001) in order to identify additional context features that it may be effective to represent in the state.

1.1 Methodology

To explore these issues we have developed a Reinforcement Learning (RL) program to learn dialogue strategies while accurate simulated users (Georgila et al., 2005a) converse with a dialogue manager. See (Singh et al., 2002; Scheffler and Young, 2001) and (Sutton and Barto, 1998) for a detailed description of Markov Decision Processes and the relevant RL algorithms.

In dialogue management we are faced with the problem of deciding which dialogue actions it is best to perform in different states. We use (RL) because it is a method of learning by *delayed* reward using *trial-and-error search*. These two properties appear to make RL techniques a good fit with the problem of automatically optimising dialogue strategies, because in task-oriented dialogue often the "reward" of the dialogue (e.g. successfully booking a flight) is not obtainable immediately, and the large space of possible dialogues for any task makes some degree of trial-and-error exploration necessary.

We use both 4-gram and 5-gram user simulations for testing and for training (i.e. train with 4-gram, test with 5-gram, and vice-versa). These simulations also simulate ASR errors since the probabilities are learned from recognition hypotheses and system behaviour logged in the COMMUNICATOR data (Walker et al., 2001) further annotated with speech acts and contexts by (Georgila et al., 2005b). Here the task domain is flight-booking, and the aim for the dialogue manager is to obtain values for the user's flight information "slots" i.e. *departure city*, *destination city*, *departure date* and *departure time*, before making a database query. We add the *dialogue moves of the last user and system turns* as context features and use these in strategy learning. We compare the learned strategies to 2 baselines: the original COMMUNICATOR systems and an RL strategy which uses only slot status features.

1.2 Outline

Section 2 contains a description of our basic experimental framework, and a detailed description of the reinforcement learning component and user simulations. Sections 3 and 4 describe the experiments and analyse our results, and in section 5 we conclude and suggest future work.

2 The Experimental Framework

Each experiment is executed using the DIPPER Information State Update dialogue manager (Bos et al., 2003) (which here is used to track and update dialogue context rather than deciding which actions to take), a Reinforcement Learning program (which determines the next dialogue action to take), and various user simulations. In sections 2.3 and 2.4 we give more details about the reinforcement learner and user simulations.

2.1 The action set for the learner

Below is a list of all the different actions that the RL dialogue manager can take and must learn to choose between based on the context:

1. An open question e.g. 'How may I help you?'
2. Ask the value for any one of slots $1...n$.
3. Explicitly confirm any one of slots $1...n$.
4. Ask for the n^{th} slot whilst implicitly confirming[1] either slot value $n-1$ e.g. 'So you want to fly from London to where?', or slot value $n+1$
5. Give help.
6. Pass to human operator.
7. Database Query.

There are a couple of restrictions regarding which actions can be taken in which states: an open question is only available at the start of the dialogue, and the dialogue manager can only try to confirm non-empty slots.

2.2 The Reward Function

We employ an "all-or-nothing" reward function which is as follows:

1. Database query, all slots confirmed: $+100$
2. Any other database query: -75

[1] Where $n = 1$ we implicitly confirm the final slot and where $n = 4$ we implicitly confirm the first slot. This action set does not include actions that ask the n^{th} slot whilst implicitly confirming slot value $n-2$. These will be added in future experiments as we continue to increase the action and state space.

3. User simulation hangs-up: -100
4. DIPPER passes to a human operator: -50
5. Each system turn: -5

To maximise the chances of a slot value being correct, it must be confirmed rather than just filled. The reward function reflects the fact that a successful dialogue manager must maximise its chances of getting the slots correct i.e. they must all be confirmed. (Walker et al., 2000) showed with the PARADISE evaluation that confirming slots increases user satisfaction.

The maximum reward that can be obtained for a single dialogue is 85, (the dialogue manager prompts the user, the user replies by filling all four of the slots in a single utterance, and the dialogue manager then confirms all four slots and submits a database query).

2.3 The Reinforcement Learner's Parameters

When the reinforcement learner agent is initialized, it is given a parameter string which includes the following:

1. Step Parameter: $\alpha = decreasing$
2. Discount Factor: $\gamma = 1$
3. Action Selection Type = *softmax* (alternative is ϵ-*greedy*)
4. Action Selection Parameter: *temperature* $= 15$
5. Eligibility Trace Parameter: $\lambda = 0.9$
6. Eligibility Trace = *replacing* (alternative is *accumulating*)
7. Initial *Q-values* = 25

The reinforcement learner updates its Q-values using the *Sarsa(λ)* algorithm (see (Sutton and Barto, 1998)). The first parameter is the step-parameter α which may be a value between 0 and 1, or specified as *decreasing*. If it is *decreasing*, as it is in our experiments, then for any given Q-value update α is $\frac{1}{k}$ where k is the number of times that the state-action pair for which the update is being performed has been visited. This kind of step parameter will ensure that given a sufficient number of training dialogues, each of the Q-values will eventually converge. The second parameter (discount factor) γ may take a value between 0 and 1. For the dialogue management problem we set it to 1 so that future rewards are taken into account as strongly as possible.

Apart from updating Q-values, the reinforcement learner must also choose the next action for the dialogue manager and the third parameter specifies whether it does this by ϵ-*greedy* or *softmax* action selection (here we have used softmax).

The fifth parameter, the eligibility trace parameter λ, may take a value between 0 and 1, and the sixth parameter specifies whether the eligibility traces are *replacing* or *accumulating*. We used *replacing* traces because they produced faster learning for the slot-filling task. The seventh parameter is for supplying the initial Q-values.

2.4 N-Gram User Simulations

Here *user simulations*, rather than real users, interact with the dialogue system during learning. This is because thousands of dialogues may be necessary to train even a simple system (here we train on up to 50000 dialogues), and for a proper exploration of the state-action space the system should sometimes take actions that are not optimal for the current situation, making it a sadistic and time-consuming procedure for any human training the system. (Eckert et al., 1997) were the first to use a user simulation for this purpose, but it was not goal-directed and so could produce inconsistent utterances. The later simulations of (Pietquin, 2004) and (Scheffler and Young, 2001) were to some extent "goal-directed" and also incorporated an *ASR error simulation*. The user simulations interact with the system via *intentions*. Intentions are preferred because they are easier to generate than word sequences and because they allow error modelling of all parts of the system, for example *ASR error modelling* and semantic errors. The user and ASR simulations must be realistic if the learned strategy is to be directly applicable in a real system.

The n-gram user simulations used here (see (Georgila et al., 2005a) for details and evaluation results) treat a dialogue as a sequence of pairs of speech acts and tasks. They take as input the $n-1$ most recent speech act-task pairs in the dialogue history, and based on n-gram probabilities learned from the COMMUNICATOR data (automatically annotated with speech acts and Information States (Georgila et al., 2005b)), they then output a user utterance as a further speech-act task pair. These user simulations incorporate the effects of ASR errors since they are built from the user utterances as they were recognized by the ASR components of the original COMMUNICATOR systems. Note that the user simulations do not provide instantiated slot values e.g. a response to provide a destination city is the speech-act task pair *"[provide info] [dest city]"*. We cannot assume that two such responses in the same dialogue refer to the same

destination cities. Hence in the dialogue manager's Information State where we record whether a slot is *empty*, *filled*, or *confirmed*, we only update from *filled* to *confirmed* when the slot value is implicitly or explicitly confirmed. An additional function maps the user speech-act task pairs to a form that can be interpreted by the dialogue manager. Post-mapping user responses are made up of one or more of the following types of utterance: (1) Stay quiet, (2) Provide 1 or more slot values, (3) Yes, (4) No, (5) Ask for help, (6) Hang-up, (7) Null (out-of-domain or no ASR hypothesis).

The quality of the 4 and 5-gram user simulations has been established through a variety of metrics and against the behaviour of the actual users of the COMMUNICATOR systems, see (Georgila et al., 2005a).

2.4.1 Limitations of the user simulations

The user and ASR simulations are a fundamentally important factor in determining the nature of the learned strategies. For this reason we should note the limitations of the n-gram simulations used here. A first limitation is that we cannot be sure that the COMMUNICATOR training data is sufficiently complete, and a second is that the n-gram simulations only use a window of n moves in the dialogue history. This second limitation becomes a problem when the user simulation's current move ought to take into account something that occurred at an earlier stage in the dialogue. This might result in the user simulation repeating a slot value unnecessarily, or the chance of an ASR error for a particular word being independent of whether the same word was previously recognised correctly. The latter case means we cannot simulate for example, a particular slot value always being liable to misrecognition. These limitations will affect the nature of the learned strategies. Different state features may assume more or less importance than they would if the simulations were more realistic. This is a point that we will return to in the analysis of the experimental results. In future work we will use the more accurate user simulations recently developed following (Georgila et al., 2005a) and we expect that these will improve our results still further.

3 Experiments

First we learned strategies with the 4-gram user simulation and tested with the 5-gram simulation, and then did the reverse. We experimented with different feature sets, exploring whether better strategies could be learned by adding limited context features. We used two baselines for comparison:

- The performance of the original COMMUNICATOR systems in the data set (Walker et al., 2001).
- An **RL baseline** dialogue manager learned using only slot-status features i.e. for each of slots $1-4$, is the slot *empty*, *filled* or *confirmed*?

We then learned two further strategies:

- **Strategy 2 (UDM)** was learned by adding the user's last dialogue move to the state.
- **Strategy 3 (USDM)** was learned by adding both the user and system's last dialogue moves to the state.

The possible system and user dialogue moves were those given in sections 2.1 and 2.4 respectively, and the reward function was that described in section 2.2.

3.1 The COMMUNICATOR data baseline

We computed the scores for the original hand-coded COMMUNICATOR systems as was done by (Henderson et al., 2005), and we call this the "HLG05" score. This scoring function is based on task completion and dialogue length rewards as determined by the PARADISE evaluation (Walker et al., 2000). This function gives 25 points for each slot which is filled, another 25 for each that is confirmed, and deducts 1 point for each system action. In this case the maximum possible score is 197 i.e. 200 minus 3 actions, (the system prompts the user, the user replies by filling all four of the slots in one turn, and the system then confirms all four slots and offers the flight). The average score for the 1242 dialogues in the COMMUNICATOR dataset where the aim was to fill and confirm only the same four slots as we have used here was 115.26. The other COMMUNICATOR dialogues involved different slots relating to return flights, hotel-bookings and car-rentals.

4 Results

Figure 1 tracks the improvement of the 3 learned strategies for 50000 training dialogues with the 4-gram user simulation, and figure 2 for 50000 training dialogues with the 5-gram simulation. They show the average reward (according to the function of section 2.2) obtained by each strategy over intervals of 1000 training dialogues.

Table 1 shows the results for testing the strategies learned after 50000 training dialogues (the baseline RL strategy, strategy 2 (UDM) and strategy 3 (USDM)). The 'a' strategies were trained with the 4-gram user simulation and tested with

	Features	Av. Score	HLG05	Filled Slots	Conf. Slots	Length
$4 \rightarrow 5$ **gram = (a)**						
RL Baseline (a)	Slots status	51.67	190.32	100	100	-9.68
RL Strat 2, UDM (a)	+ Last User DM	53.65**	190.67	100	100	-9.33
RL Strat 3, USDM (a)	+ Last System DM	54.9**	190.98	100	100	-9.02
$5 \rightarrow 4$ **gram = (b)**						
RL Baseline (b)	Slots status	51.4	190.28	100	100	-9.72
RL Strat 2, UDM (b)	+ Last User DM	54.46*	190.83	100	100	-9.17
RL Strat 3, USDM (b)	+ Last System DM	**56.24****	**191.25**	100	100	-8.75
RL Baseline (av)	Slots status	51.54	190.3	100	100	-9.7
RL Strat 2, UDM (av)	+ Last User DM	54.06**	190.75	100	100	-9.25
RL Strat 3, USDM (av)	+ Last System DM	**55.57****	**191.16**	100	100	-8.84
COMM Systems			115.26	84.6	63.7	-33.1
Hybrid RL ***	Information States		142.6	88.1	70.9	-16.4

Table 1: Testing the learned strategies after 50000 training dialogues, average reward achieved per dialogue over 1000 test dialogues. (a) = strategy trained using 4-gram and tested with 5-gram; (b) = strategy trained with 5-gram and tested with 4-gram; (av) = average; * significance level $p < 0.025$; ** significance level $p < 0.005$; *** Note: The Hybrid RL scores (here updated from (Henderson et al., 2005)) are not directly comparable since that system has a larger action set and fewer policy constraints.

the 5-gram, while the 'b' strategies were trained with the 5-gram user simulation and tested with the 4-gram. The table also shows average scores for the strategies. Column 2 contains the average reward obtained per dialogue by each strategy over 1000 test dialogues (computed using the function of section 2.2).

The 1000 test dialogues for each strategy were divided into 10 sets of 100. We carried out t-tests and found that in both the 'a' and 'b' cases, strategy 2 (UDM) performs significantly better than the RL baseline (significance levels $p < 0.005$ and $p < 0.025$), and strategy 3 (USDM) performs significantly better than strategy 2 (UDM) (significance level $p < 0.005$). With respect to average performance, strategy 2 (UDM) improves over the RL baseline by 4.9%, and strategy 3 (USDM) improves by 7.8%. Although there seem to be only negligible qualitative differences between strategies 2(b) and 3(b) and their 'a' equivalents, the former perform slightly better in testing. This suggests that the 4-gram simulation used for testing the 'b' strategies is a little more reliable in filling and confirming slot values than the 5-gram.

The 3rd column "HLG05" shows the average scores for the dialogues as computed by the reward function of (Henderson et al., 2005). This is done for comparison with that work but also with the COMMUNICATOR data baseline. Using the HLG05 reward function, strategy 3 (USDM) improves over the original COMMUNICATOR systems baseline by 65.9%. The components making up the reward are shown in the final 3 columns of table 1. Here we see that all of the RL strategies are able to fill and confirm all of the 4 slots when conversing with the simulated COMMUNICATOR users. The only variation is in the average length of dialogue required to confirm all four slots. The COMMUNICATOR systems were often unable to confirm or fill all of the user slots, and the dialogues were quite long on average. As stated in section 2.4.1, the n-gram simulations do not simulate the case of a particular user goal utterance being unrecognisable for the system. This was a problem that could be encountered by the real COMMUNICATOR systems.

Nevertheless, the performance of all the learned strategies compares very well to the COMMUNICATOR data baseline. For example, in an average dialogue, the RL strategies filled and confirmed all four slots with around 9 actions not including offering the flight, but the COMMUNICATOR systems took an average of around 33 actions per dialogue, and often failed to complete the task.

With respect to the hybrid RL result of (Henderson et al., 2005), shown in the final row of the table, Strategy 3 (USDM) shows a 34% improvement, though these results are not directly comparable because that system uses a larger action set and has fewer constraints (e.g. it can ask "how may I help you?" at any time, not just at the start of a dialogue).

Finally, let us note that the performance of the RL strategies is close to optimal, but that there is some room for improvement. With respect to the HLG05 metric, the optimal system score would be 197, but this would only be available in rare cases where the simulated user supplies all 4 slots in the

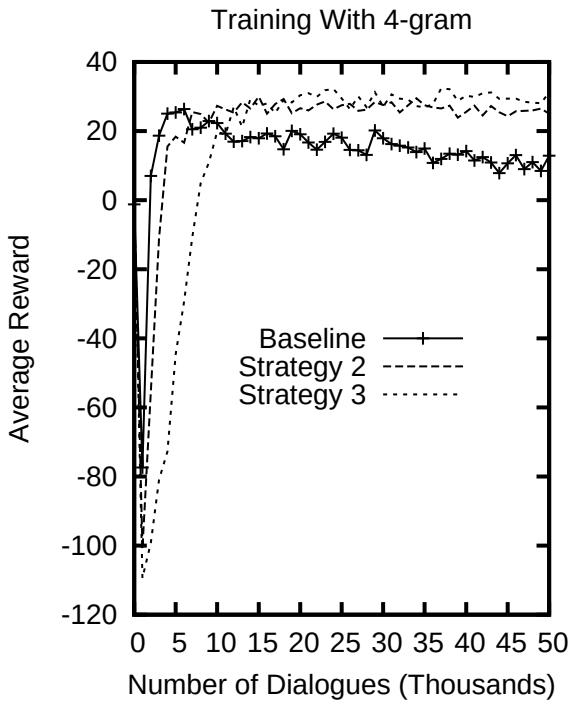

Figure 1: Training the dialogue strategies with the 4-gram user simulation

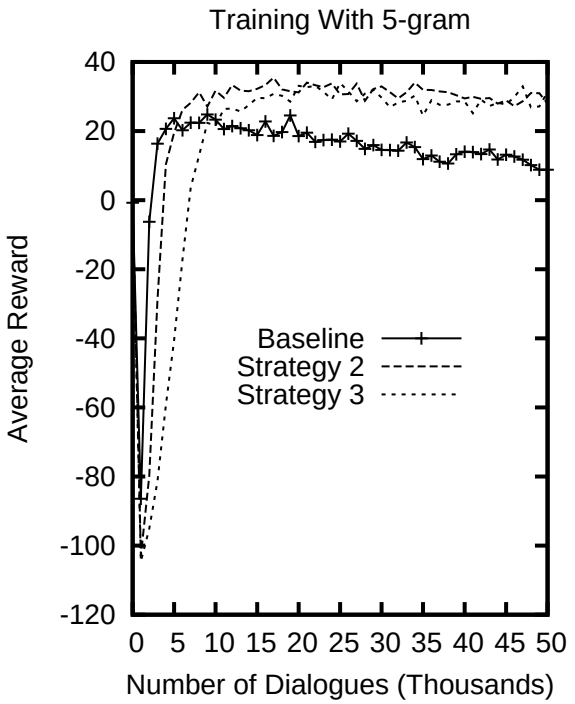

Figure 2: Training the dialogue strategies with the 5-gram user simulation

first utterance. With respect to the metric we have used here (with a -5 per system turn penalty), the optimal score is 85 (and we currently score an average of 55.57). Thus we expect that there are still further improvments that can be made to more fully exploit the dialogue context (see section 4.3).

4.1 Qualitative Analysis

Below are a list of general characteristics of the learned strategies:

1. The reinforcement learner learns to query the database only in states where all four slots have been confirmed.

2. With sufficient exploration, the reinforcement learner learns not to pass the call to a human operator in any state.

3. The learned strategies employ implicit confirmations wherever possible. This allows them to fill and confirm the slots in fewer turns than if they simply asked the slot values and then used explicit confirmation.

4. As a result of characteristic 3, which slots can be asked and implicitly confirmed at the same time influences the order in which the learned strategies attempt to fill and confirm each slot, e.g. if the status of the third slot is 'filled' and the others are 'empty', the learner learns to ask for the second or fourth slot rather than the first, since it can implicitly confirm the third while it asks for the second or fourth slots, but it cannot implicitly confirm the third while it asks for the first slot. This action is not available (see section 2.1).

4.2 Emergent behaviour

In testing the UDM strategy (2) filled and confirmed all of the slots in fewer turns on average than the RL baseline, and strategy 3 (USDM) did this in fewer turns than strategy 2 (UDM). What then were the qualitative differences between the three strategies? The behaviour of the three strategies only seems to really deviate when a user response fails to fill or confirm one or more slots. Then the baseline strategy's state has not changed and so it will repeat its last dialogue move, whereas the state for strategies 2 (UDM) and 3 (USDM) has changed and as a result, these may now try different actions. It is in such circumstances that the UDM strategy seems to be more effective than the baseline, and strategy 3 (USDM) more effective than the UDM strategy. In figure 3 we show illustrative state and learned action pairs for the different strategies. They relate to a situation where the first user response(s) in the dialogue has/have failed to fill a single slot value. NB: here 'emp' stands for 'empty' and 'fill' for 'filled' and they appear in the first four state variables, which stand for slot states. For strategy 2 (UDM), the fifth variable represents the user's last

dialogue move, and the for strategy 3 (USDM), the fifth variable represents the system's last dialogue move, and the sixth, the user's last dialogue move.

```
BASELINE STRATEGY
State:
[emp,emp,emp,emp]
Action: askSlot2


STRATEGY 2 (UDM)
State:
[emp,emp,emp,emp,user(quiet)]
Action: askSlot3

State:
[emp,emp,emp,emp,user(null)]
Action: askSlot1


STRATEGY 3 (USDM)
State:
[emp,emp,emp,emp,askSlot3,user(quiet)]
Action: askSlot3

State:
[emp,emp,emp,emp,askSlot3,user(null)]
Action: giveHelp

State:
[emp,emp,emp,emp,giveHelp,user(quiet)]
Action: askSlot3

State:
[emp,emp,emp,emp,giveHelp,user(null)]
Action: askSlot3
```

Figure 3: Examples of the different learned strategies and emergent behaviours: focus switching (for UDM) and giving help (for USDM)

Here we can see that should the user responses continue to fail to provide a slot value, the baseline's state will be unchanged and so the strategy will simply ask for slot 2 again. The state for strategy 2 (UDM) does change however. This strategy switches focus between slots 3 and 1 depending on whether the user's last dialogue move was 'null' or 'quiet' NB. As stated in section 2.4, 'null' means out-of-domain or that there was no ASR hypothesis. Strategy 3 (USDM) is different again. Knowledge of the system's last dialogue move as well as the user's last move has enabled the learner to make effective use of the 'give help' action, rather than to rely on switching focus. When the user's last dialogue move is 'null' in response to the system move 'askSlot3', then the strategy uses the 'give help' action before returning to ask for slot 3 again. The example described here is not the only example of strategy 2 (UDM) employing focus switching while strategy 3 (USDM) prefers to use the 'give help' action when a user response fails to fill or confirm a slot. This kind of behaviour in strategies 2 and 3 is emergent dialogue behaviour that has been learned by the system rather than explicitly programmed.

4.3 Further possibilities for improvement over the RL baseline

Further improvements over the RL baseline might be possible with a wider set of system actions. Strategies 2 and 3 may learn to make more effective use of additional actions than the baseline e.g. additional actions that implicitly confirm one slot whilst asking another may allow more of the switching focus described in section 4.1. Other possible additional actions include actions that ask for or confirm two or more slots simultaneously.

In section 2.4.1, we highlighted the fact that the n-gram user simulations are not completely realistic and that this will make certain state features more or less important in learning a strategy. Thus had we been able to use even more realistic user simulations, including certain additional context features in the state might have enabled a greater improvement over the baseline. Dialogue length is an example of a feature that could have made a difference had the simulations been able to simulate the case of a particular goal utterance being unrecognisable for the system. The reinforcement learner may then be able to use the dialogue length feature to learn when to give up asking for a particular slot value and make a partially complete database query. This would of course require a reward function that gave some reward to partially complete database queries rather than the all-or-nothing reward function used here.

5 Conclusion and Future Work

We have used user simulations that are n-gram models learned from COMMUNICATOR data to explore reinforcement learning of full dialogue strategies with some "high-level" context information (the user and and system's last dialogue moves). Almost all previous work (e.g. (Singh et al., 2002; Pietquin, 2004; Scheffler and Young, 2001)) has included only low-level information in state representations. In contrast, the exploration of very large state spaces to date relies on a "hybrid" supervised/reinforcement learning technique, where the reinforcement learning element has not been shown to significantly improve policies over the purely supervised case (Henderson et al., 2005).

We presented our experimental environment, the reinforcement learner, the simulated users, and our methodology. In testing with the simulated COMMUNICATOR users, the new strategies learned with higher-level (i.e. dialogue move) information in the state outperformed the low-level RL baseline (only slot status information)

by 7.8% and the original COMMUNICATOR systems by 65.9%. These strategies obtained more reward than the RL baseline by filling and confirming all of the slots with fewer system turns on average. Moreover, the learned strategies show interesting *emergent* dialogue behaviour such as making effective use of the 'give help' action and *switching focus* to different subtasks when the current subtask is proving problematic.

In future work, we plan to use even more realistic user simulations, for example those developed following (Georgila et al., 2005a), which incorporate elements of goal-directed user behaviour. We will continue to investigate whether we can maintain tractability and learn superior strategies as we add incrementally more high-level contextual information to the state. At some stage this may necessitate using a generalisation method such as linear function approximation (Henderson et al., 2005). We also intend to use feature selection techniques (e.g. CFS subset evaluation (Rieser and Lemon, 2006)) on in order to determine which contextual features this suggests are important. We will also carry out a more direct comparison with the hybrid strategies learned by (Henderson et al., 2005). In the slightly longer term, we will test our learned strategies on humans using a full spoken dialogue system. We hypothesize that the strategies which perform the best in terms of task completion and user satisfaction scores (Walker et al., 2000) will be those learned with high-level dialogue context information in the state.

Acknowledgements

This work is supported by the ESRC and the TALK project, `www.talk-project.org`.

References

John L. Austin. 1962. *How To Do Things With Words*. Oxford University Press.

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. 2003. Dipper: Description and formalisation of an information-state update dialogue system architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, Sapporo.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

Weiland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

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*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2005a. Learning User Simulations for Information State Update Dialogue Systems. In *Interspeech/Eurospeech: the 9th biennial conference of the International Speech Communication Association*.

Kallirroi Georgila, Oliver Lemon, and James Henderson. 2005b. Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations. In *Ninth Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL: DIALOR)*.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2005. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data. In *IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems,*.

Staffan Larsson and David Traum. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Eurospeech*, Rhodes,Greece.

Olivier Pietquin. 2004. *A Framework for Unsupervised Learning of Dialogue Strategies*. Presses Universitaires de Louvain, SIMILAR Collection.

Verena Rieser and Oliver Lemon. 2006. Using machine learning to explore human multimodal clarification strategies. In *Proc. ACL*.

Konrad Scheffler and Steve Young. 2001. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *NAACL-2001 Workshop on Adaptation in Dialogue Systems*, Pittsburgh, USA.

John R. Searle. 1969. *Speech Acts*. Cambridge University Press.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artifi cial Intelligence Research (JAIR)*.

Richard Sutton and Andrew Barto. 1998. *Reinforcement Learning*. MIT Press.

Marilyn A. Walker, Candace A. Kamm, and Diane J. Litman. 2000. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, 6(3).

Marilyn A. Walker, Rebecca J. Passonneau, and Julie E. Boland. 2001. Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems. In *Meeting of the Association for Computational Linguistics*, pages 515–522.