EVALUATING EFFECTIVENESS AND PORTABILITY OF REINFORCEMENT LEARNED DIALOGUE STRATEGIES WITH REAL USERS: THE TALK TOWNINFO EVALUATION

*Oliver Lemon, Kallirroi Georgila, James Henderson**

School of Informatics
Edinburgh University
olemon,kgeorgil,jhender6@inf.ed.ac.uk

ABSTRACT

We report evaluation results for real users of a learnt dialogue management policy versus a hand-coded policy in the TALK project's "TownInfo" tourist information system [1]. The learnt policy, for filling and confirming information slots, was derived from COMMUNICATOR (flight-booking) data using Reinforcement Learning (RL) as described in [2], ported to the tourist information domain (using a general method that we propose here), and tested using 18 human users in 180 dialogues, who also used a state-of-the-art hand-coded dialogue policy embedded in an otherwise identical system. We found that users of the (ported) learned policy had an average gain in perceived task completion of 14.2% (from 67.6% to 81.8% at $p < .03$), that the hand-coded policy dialogues had on average 3.3 more system turns ($p < .01$), and that the user satisfaction results were comparable, even though the policy was learned for a different domain. Combining these in a dialogue reward score, we found a 14.4% increase for the learnt policy (a 23.8% relative increase, $p < .03$). These results are important because they show a) that results for real users are consistent with results for automatic evaluation [2] of learned policies using simulated users [3, 4], b) that a policy learned using linear function approximation over a very large policy space [2] is effective for real users, and c) that policies learned using data for one domain can be used successfully in other domains. We also present a qualitative discussion of the learnt policy.

Index Terms— Natural language interfaces, Speech communication, Speech processing, User Interfaces, Learning Systems

1. INTRODUCTION

We describe and evaluate "TownInfo" [1], a multimodal dialogue system using reinforcement learning (RL) for tourist information scenarios. This is the first "Information State Update" (ISU) dialogue system to employ a learned dialogue policy, mapping complex dialogue contexts, or Information States (IS), to dialogue actions. Figure 1 shows the system's GUI, using a map and database entities developed by [5], which is used for system output only, marking the locations of various entities (hotels, restaurants, bars) on the map.

In prior work on RL for dialogue systems (e.g. [6, 7, 8, 9]), only very simple state/context representations have been used, often consisting of only the status of information "slots" (e.g. destination_city is filled with low confidence), and only specific choice points were available for learning (e.g. initiative and confirmation in [7]), whereas the policy we test [2] was learnt using linear function approximation methods over a very large state space (including e.g. speech act history) and the full range of potential dialogue actions in every state. The issues arise of how effective policies learned with this method are for real users rather than in evaluation with simulated users [2, 3], and whether evaluation results from simulated users are consistent with those for real users.

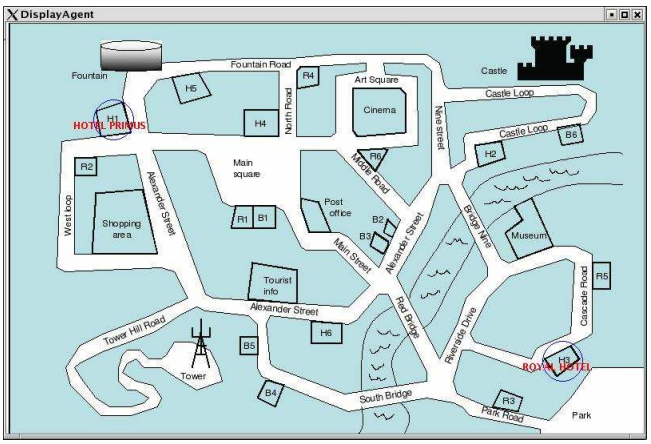

Fig. 1. The TownInfo system GUI.

We also address in this work the question of to what extent dialogue policies learnt from data gathered for one system, or family of systems, can be re-used or adapted for use in another system. We propose a general method for porting policies between domains in section 4. Our hypothesis is that the slot-filling policies learnt from our experiments with COMMUNICATOR will also be good policies for other slot-filling tasks – that is, that we have learnt "generic" slot-filling or information seeking dialogue policies.

The evaluation presented here was thus designed to answer the following questions:

- How good is the learnt strategy compared to a baseline hand-coded system strategy, for real users?
- Are results for automatic evaluation using simulated users consistent with evaluation results with real users?
- Can dialogue strategies learned in one domain be successful when they are ported to other domains?

Section 2 discusses related work, section 3 describes the system functionality, and in section 4 we describe how the dialogue policies learnt for slot filling on the COMMUNICATOR data set can be ported to the TownInfo scenarios. Section 5 presents the evaluation methodology and section 6 describes our results, including a qualitative discussion of the learnt policy (section 6.3).

*We thank the EC IST TALK Project (no. IST 507802) for funding.

2. RELATED WORK

We only know of one evaluation of a learned dialogue policy with real users, the NJFUN system [7], and we are not aware of any prior work on portability of learned dialogue policies.

In [7] 21 subjects performed 6 tasks with the NJFUN system. However, note that the baseline strategy for comparison was not a fully hand-coded policy (as we use here), but one where random choices of action were made at certain points (the "Exploratory for Initiative and Confirmation" strategy). In these conditions task completion for the learnt policy rose from 52% to 64%, with $p < .06$.

Data was then divided into the first 2 tasks ("novice users") and tasks 3-6 ("expert users") to control for learning effects. The learned strategy led to significant improvement in task completion for experts but a non-significant degradation for novices.

2.1. Automatic evaluation using simulated users

In [2] we automatically evaluated learnt dialogue management policies by running them against user simulations [3]. Both the policies and the user simulations were trained using the annotated COMMUNICATOR data described in [10] and developed on the basis of [11]. We compared our results against the performance of the original COMMUNICATOR systems, using an evaluation metric derived from the PARADISE methodology [12]. The results presented in [2] showed that the learnt policies performed better than any of the COMMUNICATOR systems, and 37% better than the best system, when tested in simulation. The important next step is to discover whether this result carries over into tests with real human users, which we present below.

3. TOWNINFO SYSTEM OVERVIEW

Two versions of the TownInfo dialogue system (hand-coded vs. learnt policy) are built around the DIPPER dialogue manager [13]. This system is used to conduct information-seeking dialogues with a user (e.g. find a particular hotel, bar, or restaurant). This allows us to compare hand-coded against learnt strategies within the same system (i.e. the other components such as the speech-synthesizer, recognizer, GUI, etc. all remain fixed).

3.1. Overview of system features

The following features are implemented:

- Interfaced to learnt or hand-coded dialogue policies
- Multiple tasks: information for hotels, bars, and restaurants
- Mixed-initiative, question accommodation/overanswering
- Open speech recognition using n-grams (HTK) [14]
- Use of dialogue plans (hand-coded version only)
- Open-initiative initial question ("How can I help you?")
- User goal/task recognition (i.e. hotels/bars/restaurants)
- Confirmations: explicit and implicit based on ASR confidence (hand-coded version only)
- Template-based NLG for presentation of database results
- Multimodal output: highlighting and naming on GUI
- Start over, quit, and help commands
- Simple user commands (e.g. "Show me the hotels")
- Logging in TALK ISU format [10]

4. PORTABILITY: MOVING BETWEEN COMMUNICATOR AND TOWNINFO DOMAINS

The learnt policies in [2] are derived from the COMMUNICATOR systems corpora [11, 10], which are in the domain of flight-booking dialogues. In [2] we reported learning a promising initial policy for COMMUNICATOR dialogues, that was evaluated only in simulation (see section 2.1), but the issue arises of how we could transfer this policy to new domains – for example the tourist information domain of TownInfo, and test its effectiveness for real users.

There are 2 main problems to be dealt with here:

- mapping between TownInfo system dialogue contexts/ information states and COMMUNICATOR information states (IS)
- mapping between the learnt COMMUNICATOR system actions and TownInfo system actions.

The learnt COMMUNICATOR policy tells us, based on a current context (or IS), what the optimal system action is (for example "request_info(dest_city)" or "explicit_confirm (depart_date)"). Obviously, in the TownInfo scenario we have no use for task types such as "destination city" and "departure date". Our method therefore is to abstract away from the particular details of the task type, but to maintain the information about dialogue moves and the *slot numbers* that are under discussion. That is, we construe the learnt COMMUNICATOR policy as a policy concerning how to fill up to 4 information slots, and then access a database and present results to the user. We also note that some slots are more important than others. For example, in COMMUNICATOR it is essential to have a destination city, otherwise no results can be found for the user. Likewise, for the TownInfo tasks, we considered the food-type, bar-type, and hotel-location to be more important to fill than the other slots. This suggests future work on investigating learned policies for partial orderings on slots via their importance for an application.

We defined the mappings shown in table 1 between COMMUNICATOR dialogue actions and TownInfo dialogue actions, for each subtask type of the TownInfo system. For example, if we are in the restaurant subtask, when the learnt policy outputs the COMMUNICATOR action "request_info(dest_city)", that dialogue move gets mapped to the TownInfo action "request_info(food_type)".

COMMUNICATOR action	TownInfo action
dest-city	food-type
depart-date	food-price
depart-time	food-location
dest-city	hotel-location
depart-date	room-type
depart-time	hotel-price
dest-city	bar-type
depart-date	bar-price
depart-time	bar-location

Table 1. Porting between domains: subtask mappings for system actions and Information States.

Note that we treat each of the 3 TownInfo subtasks (hotels, restaurants, bars) as a separate slot-filling dialogue thread, governed by COMMUNICATOR actions. This means that the very top level of the dialogue ("How may I help you?") is not governed by the learnt policy. Only when we are in a recognized subtask do we ask the COMMUNICATOR policy for the next action. Since the COMMUNICATOR policy was learnt for 4 slots, we simply "pre-fill" a slot (origin city,

since this was usually already known at the start of COMMUNICATOR dialogues) in the IS when we send it to the learned policy in order to retrieve an action.

As for the context/information state mappings, these follow the same principles. That is, we abstract over the TownInfo states to form states that are meaningful for COMMUNICATOR policies, using the same mapping (table 1). This means that, for example, a TownInfo state where `food-type` and `food-price` are filled with high confidence is mapped to a COMMUNICATOR state where `dest-city` and `depart-date` are filled with high confidence, and all other state information is identical (modulo the task names).

5. EVALUATION METHODOLOGY

We implemented both the learned policy and a hand-coded policy in the TownInfo dialogue system of [1]. The hand-coded policy was constructed using our experience in dialogue system design. It has fixed confidence score thresholds for determining types of confirmation, and allows mixed initiative, and thus is a reasonable "state-of-the-art" dialogue policy. This hand-coded policy system has a 67.6% perceived task completion score (see section 6), which is comparable to the 52% task completion score for the NJFUN baseline policy [7]. Both policies had the same (fixed) information presentation routines, and the grammar, recognizer, GUI, synthesizer, and database were equivalent across the two conditions. We evaluated the system with 18 real users, via Perceived Task Completion, Dialogue Length, and subjective evaluations for each dialogue in the two conditions.

Following the methodology of [12] we present each subject with 10 tasks (5 in each condition), controlled for learning and temporal ordering effects (i.e. for half the subjects, the learned policy was encountered first, and for the other half the hand-coded policy was encountered first).

The tasks were presented to the subjects in the following way, to prevent subjects "reading" the tasks to the system:

```
Task 1: You are on a business trip on your own.
You need to  find a hotel room  in the middle
of town. Price is no problem.
```

The users' perceived task completion (PTC) was collected like so:

```
Write the name of result that the system
presented to you (e.g. FOG BAR) here:_________

Does this item match your search? Yes/No
```

For User Preference scores, the users were then asked, for each dialogue, to evaluate the following on a 5-point Likert scale:

- "In this conversation, it was easy to get the information that I wanted."
- "The system worked the way I expected it to, in this conversation."
- "Based on my experience in this conversation, I would like to use this system regularly."

We also collected dialogue length, since longer dialogues are known to be less satisfactory for users [12] and dialogue length is a component of the reward signals used in RL for dialogue management [2, 9, 6, 7] (usually a small negative score for each system turn).

The full corpus currently consists of 180 dialogues with 18 users, and we are collecting more data with the system. The corpus is also currently being transcribed and n-best recognition hypotheses are being generated for user utterances, using ATK [14]. The final corpus will be freely released to the research community.

6. RESULTS

6.1. Perceived Task Completion (PTC) and User Preference

As shown in table 2 the results of the evaluation were a 14.2% gain (from 67.6% to 81.8%) in average perceived task completion for the learnt policy (significant at $p < 0.03$), which is a 21% relative increase in PTC. User preference is not significantly different between the two systems.

This compares favourably with the results for NJFUN [7] where task completion rose from 52% to 64%, with $p < .06$. Thus, our results are consistent with those of [7] and support the claim that learned policies can outperform hand-coded ones.

Policy	PTC (Av. %)	User pref. (Av.)	System turns (Av.)	Reward (Av.)
hand-coded	67.6	2.75	14.9	60.5
learnt	81.8	2.67	11.6	74.9

Table 2. Evaluation of the policies (18 users, 180 dialogues).

6.2. Dialogue Length and Reward

For the 180 test dialogues we found that the average number of system turns in learnt policy dialogues is 11.6 whereas for the hand-coded policy dialogues the average number of system turns is 14.9 (a significant difference at $p < 0.01$).

This means that hand-coded strategy dialogues had on average 28.4% more system turns (3.3 per dialogue) than dialogues with the learnt policy. Combined with the task completion results above, we can see that users of the learnt policy had more effective overall interactions with the system (on average shorter dialogues with greater task completion). When we compute dialogue reward by giving a 100 score for perceived task completion and -1 per system turn (as is common in RL approaches [2, 9, 6, 7]), the average reward of the learned policy dialogues is 74.9, versus 60.5 for the hand-coded policy (14.4% reward gain for the learnt strategy, significant at $p < .03$, a 23.8% relative reward increase). This is consistent with the simulated evaluation results of [2] which showed 37% more reward (than the best COMMUNICATOR system) for the learnt policy.

Overall, this evaluation shows that a policy learned using large feature spaces and action sets, using linear function approximation [2] can outperform a hand-coded policy, and it shows also that evaluations with our simulated users [2, 3, 4] are consistent with results for real users, and that a dialogue policy learned for one domain can be ported to similar domains successfully.

6.3. Qualitative description of the learnt policy

The question naturally arises of why or in what respects the learnt policy is "better" than the hand-coded one. Qualitative analysis of the results led to the following observations:

1) The learnt policy did not confirm as often as the hand-crafted policy, which was designed to implicitly confirm utterances with confidence scores over a fixed threshold and explicitly confirm utterances with scores under that threshold. This threshold was tuned

by hand in the baseline policy. Note however that the learnt policy was not optimized at all for this threshold, since the COMMUNICATOR data does not contain ASR confidence scores for training.

2) The learnt policy could choose to skip slots whereas the hand-crafted policy would insist on always filling a slot before moving to the next unfilled one (i.e. although the hand-coded policy does allow the user to skip ahead and overanswer, it always returns to unfilled slots). That feature of the hand-crafted policy caused problems for users which the system had trouble recognizing. This finding is similar to the result of [15] where a learnt policy shows (in simulation) a successful "focus switching" strategy which asks for a different slot if it is encountering problems with the current slot.

3) Our database consisted of 6 hotels, 6 bars, and 6 restaurants. Therefore if the system skipped a slot, the worst thing that could happen would be to present all hotels (or bars, or restaurants) to the user. This would be more of a problem with a much larger database. Thus, in some cases even though the learned policy did not fill all slots, users got the information they wanted since the superset of the query results, caused by unfilled slots, was still small enough to be presented. Part of our future work will thus be to investigate the effect of the database size on perceived task completion and general user satisfaction with such a strategy.

Clearly, these sorts of considerations could be implemented in a better hand-coded baseline system for comparison. However, it is policy learning itself that has revealed these strategies. In addition, we can expect that learned policies will only become better when we train them on more data and allow them to optimize on more features e.g. ASR confidence thresholds for implicit/explicit confirmation.

7. CONCLUSION AND FUTURE WORK

We reported evaluation results with 18 real users (180 dialogues) for a learned dialogue policy versus a hand-coded dialogue policy in the TALK project's "TownInfo" tourist information system [1]. The learned policy, for filling and confirming information slots, was derived from COMMUNICATOR (flight-booking) data as described in [2], ported to the tourist information domain, and tested using human users, who also used a state-of-the-art hand-coded dialogue policy embedded in an otherwise identical system. We also presented a generic method for porting learned policies between domains in similar ("slot-filling") applications.

We found that users of the (ported) learned policy had an average gain in perceived task completion of 14.2% ($p < .03$) and comparable user satisfaction results, even though the policy was learned for a different domain. For dialogue length we found that hand-coded strategy dialogues have on average 3.3 more system turns (at $p < .01$) than dialogues with the learnt policy. Combining these in a dialogue reward score, we found a 14.4% increase for the learnt policy (a 23.8% relative increase, $p < .03$). These results are important because they show that a) a policy learned using linear function approximation over a very large policy space [2] is effective for real users, b) the automatic evaluation of the learned policy [2] using simulated users [3, 4] is consistent with results for real users, and c) policies learned using dialogue data for one domain can be used successfully in other similar domains or applications.

Future work, as well as conducting a larger evaluation and using other domains/tasks, will explore the notion of similarity between domains/tasks, and under exactly what conditions learned policies are reliably portable (e.g. comparing number, type, and ordering constraints on slots, size of database etc.). We will also develop automatic evaluation metrics for dialogue policies in simulations which are strongly correlated with results from real evaluations.

8. REFERENCES

[1] Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle, "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system," in *Proceedings of EACL*, 2006.

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

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

[4] Kallirroi Georgila, James Henderson, and Oliver Lemon, "User simulation for spoken dialogue systems: Learning and evaluation," in *Interspeech/ICSLP*, 2006, p. (to appear).

[5] Matthew Stuttle, Jason Williams, and Steve Young, "A framework for dialog systems data collection using a simulated ASR channel," in *ICSLP 2004*, Jeju, Korea, 2004.

[6] E. Levin and R. Pieraccini, "A stochastic model of computer-human interaction for learning dialogue strategies," in *Proc. Eurospeech*, 1997.

[7] Diane Litman, Micheal Kearns, Satinder Singh, and Marilyn Walker, "Automatic optimization of dialogue management," in *Proc. COLING*, 2000.

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

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

[10] Kallirroi Georgila, Oliver Lemon, and James Henderson, "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)*, 2005.

[11] M. Walker, A. Rudnicky, Aberdeen J., E. Bratt, Garofolo J., H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, R. Prasad, Roukos S., G. Sanders, S. Seneff, D. Stallard, and S. Whittaker, " DARPA Communicator Evaluation: Progress from 2000 to 2001," in *Proc. ICSLP*, 2002.

[12] Marilyn Walker, Candace Kamm, and Diane Litman., "Towards Developing General Models of Usability with PARADISE," *Natural Language Engineering*, vol. 6, no. 3, 2000.

[13] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka, "DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture," in *4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, 2003, pp. 115–124.

[14] Steve Young, *ATK: an application toolkit for HTK, version 1.4*, 2004.

[15] Matthew Frampton and Oliver Lemon, "Learning more effective dialogue strategies using limited dialogue move features," in *ACL*, 2006.