An ISU Dialogue System Exhibiting Reinforcement Learning of Dialogue Policies: Generic Slot-filling in the TALK In-car System

Oliver Lemon, Kallirroi Georgila, and **James Henderson** **Matthew Stuttle**
School of Informatics
University of Edinburgh
olemon@inf.ed.ac.uk

Dept. of Engineering
University of Cambridge
mns25@cam.ac.uk

Abstract

We demonstrate a multimodal dialogue system using reinforcement learning for in-car scenarios, developed at Edinburgh University and Cambridge University for the TALK project[1]. This prototype is the first "Information State Update" (ISU) dialogue system to exhibit reinforcement learning of dialogue strategies, and also has a fragmentary clarification feature. This paper describes the main components and functionality of the system, as well as the purposes and future use of the system, and surveys the research issues involved in its construction. Evaluation of this system (i.e. comparing the baseline system with handcoded vs. learnt dialogue policies) is ongoing, and the demonstration will show both.

1 Introduction

The in-car system described below has been constructed primarily in order to be able to collect data for Reinforcement Learning (RL) approaches to multimodal dialogue management, and also to test and further develop learnt dialogue strategies in a realistic application scenario. For these reasons we have built a system which:

- contains an interface to a dialogue strategy learner module,
- covers a realistic domain of useful "in-car" conversation and a wide range of dialogue phenomena (e.g. confirmation, initiative, clarification, information presentation),
- can be used to complete measurable tasks (i.e. there is a measure of successful and unsuccessful dialogues usable as a reward signal for Reinforcement Learning),
- logs all interactions in the TALK data collection format (Georgila et al., 2005).

[1]This research is supported by the TALK project (European Community IST project no. 507802), http://www.talk-project.org

In this demonstration we will exhibit the software system that we have developed to meet these requirements. First we describe the domain in which the dialogue system operates (an "in-car" information system). Then we describe the major components of the system and give examples of their use. We then discuss the important features of the system in respect to the dialogue phenomena that they support.

1.1 A System Exhibiting Reinforcement Learning

The central motivation for building this dialogue system is as a platform for Reinforcement Learning (RL) experiments. The system exhibits RL in 2 ways:

- It can be run in online learning mode with real users. Here the RL agent is able to learn from successful and unsuccessful dialogues with real users. Learning will be much slower than with simulated users, but can start from an already learnt policy, and slowly improve upon that.
- It can be run using an already learnt policy (e.g. the one reported in (Henderson et al., 2005; Lemon et al., 2005), learnt from COMMUNICATOR data (Georgila et al., 2005)). This mode can be used to test the learnt policies in interactions with real users.

Please see (Henderson et al., 2005) for an explanation of the techniques developed for Reinforcement Learning with ISU dialogue systems.

2 System Overview

The baseline dialogue system is built around the DIPPER dialogue manager (Bos et al., 2003). This system is initially used to conduct information-seeking dialogues with a user (e.g. find a particular hotel and restaurant), using hand-coded dialogue strategies (e.g. always use implicit confirmation, except when ASR confidence is below 50%, then use explicit confirmation). We have then modified the DIPPER dialogue manager so that it can consult learnt strategies (for example strategies learnt from the 2000 and 2001 COMMUNICATOR data (Lemon et al., 2005)), based on its

current information state, and then execute dialogue actions from those strategies. This allows us to compare hand-coded against learnt strategies within the same system (i.e. the other components such as the speech-synthesiser, recogniser, GUI, etc. all remain fixed).

2.1 Overview of System Features

The following features are currently implemented:

- use of Reinforcement Learning policies or dialogue plans,
- multiple tasks: information seeking for hotels, bars, and restaurants,
- overanswering/ question accommodation/ user-initiative,
- open speech recognition using n-grams,
- confirmations - explicit and implicit based on ASR confidence,
- fragmentary clarifications based on word confidence scores,
- multimodal output - highlighting and naming entities on GUI,
- simple user commands (e.g. "Show me all the indian restaurants"),
- dialogue context logging in ISU format (Georgila et al., 2005).

3 Research Issues

The work presented here explores a number of research themes, in particular: using learnt dialogue policies, learning dialogue policies in online interaction with users, fragmentary clarification, and reconfigurability.

3.1 Moving between Domains: COMMUNICATOR and In-car Dialogues

The learnt policies in (Henderson et al., 2005) focussed on the COMMUNICATOR system for flight-booking dialogues. There we reported learning a promising initial policy for COMMUNICATOR dialogues, but the issue arises of how we could transfer this policy to new domains – for example the in-car domain.

In the in-car scenarios the genre of "information seeking" is central. For example the SACTI corpora (Stuttle et al., 2004) have driver information requests (e.g. searching for hotels) as a major component.

One question we address here is 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 other systems. We conjecture 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 are learning "generic" slot-filling or information seeking dialogue policies. In section 5 we describe how the dialogue policies learnt for slot filling on the COMMUNICATOR data set can be abstracted and used in the in-car scenarios.

3.2 Fragmentary Clarifications

Another research issue we have been able to explore in constructing this system is the issue of generating fragmentary clarifications. The system can be run with this feature switched on or off (off for comparison with COMMUNICATOR systems). Instead of a system simply saying "Sorry, please repeat that" or some such similar simple clarification request when there is a speech recognition failure, we were able to use the word confidence scores output by the ATK speech recogniser to generate more intelligent fragmentary clarification requests such as "Did you say a cheap chinese restaurant?". This works by obtaining an ASR confidence score for each recognised word. We are then able to try various techniques for clarifying the user utterance. Many possibilities arise, for example: explicitly clarify only the highest scoring content word below the rejection threshold, or, implicitly clarify all content words and explicitly clarify the lowest scoring content word.

The current platform enables us to test alternative strategies, and develop more complex ones.

4 The "In-car" Scenario

The scenario we have designed the system to cover is that of information seeking about a town, for example its hotels, restaurants, and bars. We imagine a driver who is travelling towards this town, or is already there, who wishes to accomplish relatively complex tasks, such as finding an italian restaurant near their hotel, or finding all the wine bars in town, and so on. The driver/user should be able to specify queries using natural dialogue, and will receive system output that is a mixture of spoken and graphical information (e.g. a description of an item and a map showing its location).

The example town is taken from the (Stuttle et al., 2004) corpus collection materials, and contains a number of hotels, bars, restaurants, and tourist attractions.

The user should be able to get information on a range of locations in the town, and the dialogue system will be used to specify and refine the user queries, as well as to present information to the user. See the example dialogue in table 1.

We now describe the dialogue system components.

5 Component-level Description

This section describes the components of the baseline in-car dialogue system. Communication between components is handled by OAA's asynchronous hub architecture (Cheyer and Martin, 2001). The major components are:

- DIPPER (Bos et al., 2003) ISU dialogue manager,
- Dialogue Policy Reinforcement Learner Agent[2],
- ATK for speech recognition,
- Festival speech synthesiser,

[2]This is written in Python and has an OAA wrapper in C.

	Utterance	Feature
S	How can I help you?	Open initiative initial question
U	I'm looking for a cheap chinese restaurant.	User initiative
S	Would you like something central?	Intention recognition
U	Yes.	
S	The Dragon Way is an affordable cantonese restaurant located on Main Street **[displayed on map]**.	Multimodal presentation
S	Are you happy with those options?	
U	Yes, and I also want a double room.	
S	Would you like something expensive, cheap, or reasonably priced?	Intention recognition
U	A luxury hotel, in the center of town.	
S	There are 2 hotel options meeting your requirements. The Royal Hotel **[displayed on map]** is a renovated nineteenth century palace ... The Hotel Primus **[displayed on map]** has big, bright, clean rooms ... Which hotel option would you like?	Multimodal presentation
U	The second one please.	
S	The phone number of the Hotel Primus is 2094-227.	
S	Would you like any further information?	
U	I want to find a jazz bar.	
S	I'm sorry, what kind of bar are you looking for?	Intention recognition, Fragmentary clarification
U	A jazz bar.	
S	A jazz bar. Would you like something central?	Implicit confirmation

Table 1: Example dialogue, showing system features

- Multimodal Map interface (a java OAA agent),
- Database agent (java OAA wrapper to MySQL).

5.1 Dialogue Policy Learner Agent

This agent acts as an interface between the DIPPER dialogue manager and the system simulation based on RL. In particular it has the following solvable:
`callRLsimulation(IS_file_name, conversational domain, speech act, task, result).`
The first argument is the name of the file that contains all information about the current information state, which is required by the RL algorithm to produce an action. The action returned by the RL agent is a combination of `conversational domain`, `speech act`, and `task`. The last argument shows whether the learnt policy will continue to produce more actions or release the turn. When run in online learning mode the agent not only produces an action when supplied with a state, but at the end of every dialogue it uses the reward signal to update its learnt policy. The reward signal is defined in the RL agent, and is currently a linear combination of task success metrics combined with a fixed penalty for dialogue length (see (Henderson et al., 2005)).

This agent can be called whenever the system has to decide on the next dialogue move. In the original hand-coded system this decision is made by way of a dialogue plan (using the "deliberate" solvable). The RL agent can be used to drive the entire dialogue policy, or can be called only in certain circumstances. This makes it usable for whole dialogue strategies, but also, if desired, it can be targetted only on specific dialogue management decisions (e.g. implicit vs. explicit confirmation, as was done by (Litman et al., 2000)).

One important research issue is that of tranferring learnt strategies between domains. We learnt a strategy for the COMMUNICATOR flight booking dialogues (Lemon et al., 2005; Henderson et al., 2005), but this is generated by rather different scenarios than the in-car dialogues. However, both are "slot-filling" or information-seeking applications. We defined a mapping (described below) between the states and actions of both systems, in order to construct an interface between the learnt policies for COMMUNICATOR and the in-car baseline system.

5.2 Mapping between COMMUNICATOR and the In-car Domains

There are 2 main problems to be dealt with here:

- mapping between in-car system information states and COMMUNICATOR information states,
- mapping between learnt COMMUNICATOR system actions and in-car system actions.

The learnt COMMUNICATOR policy tells us, based on a current IS, what the optimal system action is (for example `request_info(dest_city)` or `acknowledgement`). Obviously, in the in-car 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 (ordered) informational slots, and then access a database and present results to the user. We also note that some slots are more essential 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 in-car tasks, we consider the food-type, bar-type, and hotel-location to be more important to fill than the other slots. This suggests a partial ordering on slots via their importance for an application.

In order to do this we define the mappings shown in table 2 between COMMUNICATOR dialogue actions and in-car dialogue actions, for each sub-task type of the in-car system.

COMMUNICATOR action	In-car 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 2: Action mappings

Note that we treat each of the 3 in-car sub-tasks (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 recognised task do we ask the COMMUNICATOR policy for the next action. Since the COMMUNICATOR policy is learnt for 4 slots, we "prefill" a slot[3] in the IS when we send it to the Dialogue Policy Learner Agent in order to retrieve an action.

As for the state mappings, these follow the same principles. That is, we abstract from the in-car states to form states that are usable by COMMUNICATOR . This means that, for example, an in-car 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). Note that in a future version of the in-car system where task switching is allowed we will have to maintain a separate view of the state for each task.

In terms of the integration of the learnt policies with the DIPPER system update rules, we have a system flag which states whether or not to use a learnt policy. If this flag is present, a different update rule fires when the system determines what action to take next. For example, instead of using the `deliberate` predicate to access a dialogue plan, we instead call the Dialogue Policy Learner Agent via OAA, using the current Information State of the system. This will return a dialogue action to the DIPPER update rule.

In current work we are evaluating how well the learnt policies work for real users of the in-car system.

6 Conclusions and Future Work

This report has described work done in the TALK project in building a software prototype baseline "Information State Update" (ISU)-based dialogue system in the in-car domain, with the ability to use dialogue policies derived from machine learning and also to perform online learning through interaction. We described the scenarios, gave a component level description of the software, and a feature level description and example dialogue.

Evaluation of this system (i.e. comparing the system with hand-coded vs. learnt dialogue policies) is ongoing. Initial evaluation of learnt dialogue policies (Lemon et al., 2005; Henderson et al., 2005) suggests that the learnt policy performs at least as well as a reasonable hand-coded system (the TALK policy learnt for COMMUNICATOR dialogue management outperforms all the individual hand-coded COMMUNICATOR systems).

The main achievements made in designing and constructing this baseline system have been:

- Combining learnt dialogue policies with an ISU dialogue manager. This has been done for online learning, as well as for strategies learnt offline.
- Mapping learnt policies between domains, i.e. mapping Information States and system actions between DARPA COMMUNICATOR and in-car information seeking tasks.
- Fragmentary clarification strategies: the combination of ATK word confidence scoring with ISU-based dialogue management rules allows us to explore word-based clarification techniques.

References

J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, Sapporo.

A. Cheyer and D. Martin. 2001. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1):143–148.

K. Georgila, O. Lemon, and J. Henderson. 2005. 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'05*.

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

O. Lemon, K. Georgila, J. Henderson, M. Gabsdil, I. Meza-Ruiz, and S. Young. 2005. D4.1: Integration of Learning and Adaptivity with the ISU approach. Technical report, TALK Project.

D. Litman, M. Kearns, S. Singh, and M. Walker. 2000. Automatic optimization of dialogue management. In *Proc. COLING*.

M. Stuttle, J. Williams, and S. Young. 2004. A framework for dialog systems data collection using a simulated ASR channel. In *ICSLP 2004*, Jeju, Korea.

[3] We choose "orig_city" because it is the least important and is already filled at the start of many COMMUNICATOR dialogues.