USING DIALOGUE ACTS TO LEARN BETTER REPAIR STRATEGIES FOR SPOKEN DIALOGUE SYSTEMS

Matthew Frampton

Center for the Study of Language and Information
Stanford University
frampton@stanford.edu

Oliver Lemon *

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

ABSTRACT

Repair or error-recovery strategies are an important design issue in Spoken Dialogue Systems (SDSs) - how to conduct the dialogue when there is no progress (e.g. due to repeated ASR errors). Nearly all current SDSs use hand-crafted repair rules, but a more robust approach is to use Reinforcement Learning (RL) for data-driven dialogue strategy learning. However, as well as usually being tested only in simulation, current RL approaches use small state spaces which do not contain linguistically motivated features such as "Dialogue Acts" (DAs). We show that a strategy learned with DA features outperforms hand-crafted and slot-status strategies when tested with real users ($+9\%$ average task completion, $p < 0.05$). We then explore how using DAs produces better repair strategies e.g. focus-switching. We show that DAs are useful in deciding both when to use a repair strategy, and which one to use.

Index Terms— Natural language interfaces, Adaptive systems, Speech processing, Cooperative systems

1. INTRODUCTION

Using Reinforcement Learning (RL) [1] to learn dialogue strategies for Spoken Dialogue Systems (SDSs) was initially proposed by [2]. RL involves learning a series of actions to take in different states so as to maximize long-term reward. Applied to SDSs, actions correspond to system actions, states represent the dialogue context, and the reward function rewards favourable dialogue outcomes e.g. task completion, short length, high user satisfaction. In earlier research where strategies were learned for information seeking/slot-filling SDSs, the state contained only slot-status information e.g. whether a slot was filled/confirmed, but more recently [3] and [4] have explored adding Dialogue Acts (DAs). DAs are linguistic representations of the communicative intent of a user or system utterance, e.g. "provide_info(hotel_type(luxury))" or "request_info(origin)". Our DA taxonomy is based on the DATE [5] scheme used for the COMMUNICATOR corpus. Using a reward function based on task completion and dialogue length [4] learned and tested strategies with n-gram simulations derived from COMMUNICATOR data [6] and found that a small amount of DA history improved learned strategy performance. The DA-strategies often differed from the "slot-status only" strategy when the user failed to fill/confirm a slot value (e.g. due to an ASR error). Whereas the latter would always repeat its last question/ confirmation, the former sometimes employed alternative repair strategies e.g. switching focus to another slot or giving help. These repair strategies improved performance - they were more effective at getting the dialogue back on track and so achieved task completion in fewer turns.

This prior work [4] then suggests that previous DAs can be used to learn better repair strategies, but outstanding issues remain which we address here. First, Section 2 will describe how we learn dialogue strategies and provide initial analysis, before Section 3 describes an experiment which tests whether the DA-strategies also perform better with real users of SDSs. Given the positive results, we move on to further simulation experiments which investigate why the DA-strategies are better. In Section 4, we test whether the DAs are only proving useful for learning repair strategies. Section 5 then describes an experiment which investigates the importance of DAs in choosing *which* repair strategy to apply.

2. LEARNING THE DIALOGUE STRATEGIES

We built an RL program which uses the Sarsa(λ) algorithm (see [1]) to learn a dialogue strategy as it interacts with a stochastic user simulation via a dialogue manager. For learning and testing here and in Sections 4 and 5, the reward function gives +100 if all of the slots are confirmed at the end of a dialogue, and -5 for each system turn. The user simulations are n-gram models (n=4 and n=5) learned from the COMMUNICATOR data that output new user DA(s) based on the DAs of the last n-1 turns [6]. We learned three-slot strategies with the 4-gram simulation and tested them with the 5-gram, and vice-versa (the "TownInfo" system [7] used in our real user experiment, (see Section 3), has three information slots). The following state representations were used:

*This work is partially funded by the EPSRC (grant number EP/E019501/1) and the EC FP7 project "CLASSiC" (ICT-216594).

- Slot-Status Strategy: Only the slot-status features, where each can take the value "empty", "filled" or "confirmed"
- DA1-Strategy: The slot-status features + the DA(s) of the last user turn.
- DA2-Strategy: The slot-status features + the DAs of the last system and user turns.

DAs of turns further back than the last system turn were beyond the context window to which the 4-gram user simulation can be sensitive, and so could not have improved the learned strategy. The system DA set, (which is the same throughout the paper), includes a mixed-initiative open question i.e. "How may I help you?", questions for each slot, explicit confirmations, combinations of implicit confirmations and questions, giving help, and querying the database for suitable options (this terminates the dialogue). The reinforcement learner can only choose the open question on its first turn in the dialogue, and it can only confirm non-empty slots.

After 50000 training dialogues [1], each strategy was tested over 10 sets of 100 dialogues. Since the simulations failed to simulate unobtainable slot-values, in testing here, and in the experiments of Sections 4 and 5, the strategies were always able to confirm all of the slots. However due to shorter dialogue length, like [4], we found that DA1 outperformed the Slot-Status Strategy ($p < 0.05$), and DA2 outperformed DA1 ($p < 0.05$). On analysis, the DA-strategies only seemed to differ from the Slot-Status Strategy when the user failed to fill/confirm a slot . Whereas the Slot-Status Strategy would always repeat its last question/confirmation, the DA-strategies used additional repair strategies: "switching focus" (asking/confirming a different unfilled/unconfirmed slot), giving help, and "backtracking" (re-asking a filled slot). NB. See Table 1 for examples of the first two strategies. Although it repeats its previous question when the very first user turn fails to fill a slot value, DA2 often uses one of the three new repair strategies: it uses switching focus more than giving help and backtracking and tends to use the latter two if the dialogue remains stalled after first switching focus. Other usages are when two of the three slots are confirmed and hence switching focus is impossible. These new repair strategies then apparently improved performance, and we will analyse this futher below.

3. TESTING ON REAL USERS

We next investigated whether learning with DAs had produced strategies which also perform better with real users. Like [8], although our strategies were learned using COMMUNICATOR users (in the flight-booking domain), we tested on real users in a tourist information domain using the "TownInfo" SDS [7]. To do this, we treat the learned policy as an abstract policy for filling user information slots, regardless of their specific type (e.g. destination city vs. restaurant cuisine).

[1] For DA2, (the largest state-action space), approx. 6000 state-action pairs were visited with the 4-gram simulation, and approx. 6000 with the 5-gram.

Speaker	System state/User utt & (DA)	Sys action
System:	[e,e,e,none,none]	askSlot1
User:	"..." (quiet)	
System:	[e,e,e,askSlot1,quiet]	askSlot1
User:	"[ASR rejection]" (null)	
System:	[e,e,e,askSlot1,null]	askSlot3
User:	"26th of June" (giveSlot3)	
System:	[e,e,f,askSlot3,giveSlot3]	IC3_askSlot2
User:	"..." (quiet)	
System:	[e,e,c,IC3_ask2,quiet]	askSlot1
User:	"..." (quiet)	
System:	[e,e,c,askSlot1,quiet]	giveHelp
User:	"[ASR rejection]" (null)	
System:	[e,e,c,giveHelp,null]	askSlot1
User:	"New York" (giveSlot1)	

Table 1. Example exchanges in a learned DA2-strategy dialogue showing different repair strategies: re-asking, switching focus, and giving help. In the system's states, the first 3 positions are slot-status ('e" = empty, "f" = filled, "c" = confirmed), and the next 2 positions are the DAs of the last system and user turns ("IC3" = implicit confirmation of slot 3.) In COMMUNICATOR, slot 1 = departure city, slot 2 = destination city, and slot 3 = depart date.

3.1. Methodology

In the first phase of a "TownInfo" dialogue, the system must ascertain whether the user is interested in a hotel, bar, or restaurant, and then during the second slot-filling phase, try to elicit information from the user to fill and confirm slots for *type*, *location*, and *price*. In the final phase, the system retrieves suitable options from its database, and uses its GUI to highlight them on a map. We left the hand-crafted strategies used in the first and final stages unchanged, and tested three different strategies in the slot-filling phase - the Slot-Status Strategy, the DA2-Strategy, and a state-of-the-art mixed-initiative hand-crafted strategy. Implementing the strategies involved mapping between COMMUNICATOR and "TownInfo" slots types as described in [8]. The hand-crafted strategy attempts to fill/confirm the slots in the default order $1-3$ (overanswering is allowed), and does not make a database query until all slots are confirmed. Since the DA-strategy had been learned in a different domain, if it was successful here then it could be considered an effective generic slot-filling strategy. Each of 11 subjects was given a questionnaire which contained 15 tasks to attempt, (5 for each of the 3 strategies), and learning and temporal ordering effects were controlled for. The slot-filling strategy was the only part of the system that was varied. Evaluation measures were collected for dialogue length and Perceived Task Completion (PTC).

3.2. Results and Analysis

As displayed in Table 2, PTC for the DA-Strategy was higher than for both the Hand-crafted and Slot-Status Strategy ($+9\%$, $p < 0.05$) [2]. The average number of system turns per dialogue for the DA-Strategy was fewer than for the Slot-Status Strategy ($p < 0.05$).

Strategy	PTC %	Av. Sys turns
DA2	90.91**	7.95*
Hand-crafted	81.82	8.46
Slot-status	81.82	8.98

Table 2. PTC = Perceived Task Completion; ** = significantly better than both Hand-crafted and S-S ($p < 0.05$), * = significantly better than S-S strategy ($p < 0.05$).

These experiments certainly emphasized why re-asking or re-attempting a confirmation is often a poor repair strategy. When the last user turn had failed to fill/confirm a slot-value it was usually due to one or more ASR errors. The repetition then employed by the Slot-Status and hand-crafted strategies was very likely to frustrate the user, eliciting irritated/hyperarticulate speech which caused further ASR errors, and so longer dialogues and lower task completion.

The positive result justifies training and testing new strategies with the n-gram simulations (see also [8]) in order to address outstanding issues. For instance, do the DAs really only add value in terms of learning repair strategies? With respect to the repair strategies, are the DAs only useful for indicating that the slot-status features are unchanged and hence that a repair strategy is required, or do they also affect *which* repair strategy is best to use? Perhaps they do affect which is best to use but nonetheless, finding the optimal repair strategy is very simple because *any* "sensible" avoidance of repetition will always be optimal. We consider any of the repair strategies that emerged from the RL to be "sensible", e.g. having asked for a slot-value on the last turn, switching focus, giving help or backtracking would be "sensible". Recall that the RL dialogue manager cannot attempt to confirm empty slots - clearly not a sensible way to avoid repetition.

4. ARE DIALOGUE ACTS ONLY USEFUL FOR REPAIR STRATEGIES?

We first investigated whether DAs only improve the learned strategy through repair strategies - perhaps DAs also make other less noticeable improvements. To explore this, we learned Strategies DA1$^{\beta}$ and DA2$^{\beta}$ for which the DA features were changed to only take a DA value when the slot-status features are unchanged. Hence DAs were ignored where progress was being made in the dialogue.

[2] By chance, both the S-S and Hand-crafted systems achieved PTC in 45 out of 55 dialogues (81.82%)

4.1. Results

As displayed in Table 3, there was no significant difference in the performance of Strategies DA1 and DA1$^{\beta}$, or DA2 and DA2$^{\beta}$. Hence ignoring DAs in contexts where progress is being made in the dialogue has no significant impact on the learned strategy's performance - the value of the DAs here seems to be entirely in learning repair strategies.

Strategy	Av. Reward	Conf. Slots	Sys. Turns
DA2	59.46	100	8.11
DA2$^{\beta}$	59.40*	100	8.12*
DA1$^{\beta}$	58.86	100	8.23
DA1	58.56	100	8.29

Table 3. Each strategy trained for 50000 dialogues with 4-gram, then tested for 1000 with 5-gram, and vice versa. Scores are average of the two tests. * = improvement ($p < 0.05$) over strategy beneath in list.

5. THE IMPORTANCE OF DIALOGUE ACTS IN CHOOSING A REPAIR STRATEGY

DAs indicate when slot-status features are unchanged, and hence a repair strategy is required, but do they also affect *which* repair strategy is best to apply? That the top-performing DA strategies use different repair strategies where only the DAs differ (see Table 1) suggests they do. To investigate further, we first learned a new strategy using slot-status features and a Slot-Features Unchanged (SFU) feature which records whether the former are unchanged by the last user turn. Recall that the real user tests highlighted that repetition can be a poor repair strategy. Hence there is the possiblility that DAs do affect which repair strategy should be used, but choosing the best repair strategy is still very simple because any "sensible" avoidance of repetition will always be optimal, ("sensible" was defined in Section 3.3.). Hence we also tested three strategies which follow DA2 until the slot-status features are unchanged, and then they always take a sensible action which is different from the last system action. The first switches focus to an unfilled/unconfirmed slot (SF) [3], while assuming the slot-status features continue to be unchanged, the second alternates betweeen first giving help and then re-attempting the original question/confirmation (HR), and the third between giving help and then switching focus (HSF).

5.1. Results

The testing results are displayed in Table 4. They show that the new SFU learned strategy and the strategies which always use a sensible "non-repeating" repair strategy but otherwise

[3] If there are no unfilled/unconfirmed slots to switch focus to, the strategy continues to follow DA2.

follow DA2 all perform significantly worse than DA2. Hence recent DAs are useful for choosing which repair strategy to apply, and it is not the case that any sensible repair strategy which avoids repetition will necessarily be optimal.

Strategy	Av. Reward	Conf. Slots	Sys. Turns
DA2	59.46*	100	8.11*
DA2+HR	54.15	100	9.17
SFU	53.47	100	9.31
DA2+HSF	53.22	100	9.36
DA2+SF	52.01	100	9.60

Table 4. Strategies, (except for repair parts - SF, HR and HSF), trained for 50000 dialogues with 4-gram simulation, then tested for 1000 with 5-gram, and vice versa. Scores are average of the two tests. * = improvement ($p < 0.005$).

6. CONCLUSIONS

We used RL with COMMUNICATOR n-gram user simulations [6] to learn dialogue strategies. Like [4], we found that state spaces using DAs in addition to slot-status information produced strategies which performed better in testing with the simulations. Since the DA-strategies only seemed to differ when the user failed to fill/confirm a slot value, this seemed to be why. Whereas the Slot-Status Strategy would always repeat its last question/confirmation, the DA-strategies often employed alternative repair strategies: switching focus (asking/confirming a different unfilled/ unconfirmed slot), giving help, and backtracking (re-asking a filled slot).

In the first of three further experiments, we then ported a DA strategy, the Slot-Status strategy and a state-of-the-art hand-coded strategy to the tourist information domain, and tested on 11 real users (165 dialogues) with the "TownInfo" SDS [7]. The DA-strategy outperformed the other two strategies in terms of task completion ($+9\%$ average, $p < 0.05$), and dialogue length ($p < 0.05$ v. Slot-status). This then confirmed that the DAs had produced a strategy which was more effective for real users, and justified using the simulations for further analysis. Analysing the real user experiments emphasized why repetition is often not an optimal repair stategy - repetition frequently frustrated the user, eliciting hyperarticulated/irritated speech which caused more ASR errors, and so longer dialogues and lower task completion.

In the second experiment, when tested with the n-gram simulations, strategies learned with DAs only when the slot-status features were unchanged, performed as well as the equivalent original DA strategies. Hence the DAs only seemed to improve the learned strategy with regard to repair. In the final experiment, worse performance than DA-strategies ($+ > 1$ system turn on average, $p < 0.05$) is obtained firstly by a strategy learned with a feature that records whether the slot-status features are unchanged, and secondly by strategies which follow the best DA-strategy until the slot-status features are unchanged, whereupon they always use the same sensible "non-repeating" repair strategies e.g switching focus. The first result shows that DAs improve the learned strategy not only because they indicate when a repair stategy is required, but also because they affect which is best to use (see Table 1 for examples). Given this, the second result shows that finding the best repair strategy is not as simple as choosing any sensible action which avoids repeating the last system action.

Future work will involve addressing the outstanding issue of how much dialogue history is relevant. A possible methodology is to apply feature selection e.g. CFS [9] to the COMMUNICATOR data. If this suggests that we should ideally use more dialogue history in the RL, to do so would then require a user simulation whose behaviour is sensitive in a realistic manner to this additional dialogue history. These approaches will ultimately lead to a data-driven methodology for the design and automatic optimization of robust SDSs.

7. REFERENCES

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, The MIT Press. Cambridge, Massachusetts, USA, 1998.

[2] E. Levin, R. Pieraccini, and W. Eckert, "Using Markov Decision Processes for learning dialogue strategies," in *Proc. ICASSP*, 1998.

[3] J. Henderson, O. Lemon, and K. Georgila, "Hybrid reinforcement / supervised learning of dialogue policies from fixed datasets," *Computational Linguistics*, 2008, (to appear).

[4] M. Frampton and O. Lemon, "Learning more effective dialogue strategies using limited dialogue move features," in *Proc. ACL*, 2006.

[5] M. Walker and R. Passonneau, "DATE: A Dialogue Act Tagging Scheme for Evaluation of Spoken Dialogue Systems," in *Proc. of HLT Conference*, 2001.

[6] K. Georgila, J. Henderson, and O. Lemon, "Learning User Simulations for Information State Update Dialogue Systems," in *Eurospeech*, 2005.

[7] O. Lemon, K. Georgila, J. Henderson, and M. Stuttle, "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system," in *Proc. of EACL*, 2006.

[8] O. Lemon, K. Georgila, and J. Henderson, "Evaluating effectiveness and portability of reinforcement learned strategies," in *Spoken Language Technology*, 2006.

[9] M. Hall, *Correlation-based Feature Selection For Machine Learning*, Ph.D. thesis, Waikato University, 1999.