Dialogue Policy Learning for combinations of Noise and User Simulation: transfer results

Oliver Lemon
Edinburgh University
olemon@inf.ed.ac.uk

Xingkun Liu
Edinburgh University
xliu4@inf.ed.ac.uk

Abstract

Once a dialogue strategy has been learned for a particular set of conditions, we need to know how well it will perform when deployed in different conditions to those it was specifically trained for, i.e. how robust it is in *transfer* to different conditions. We first present novel learning results for different ASR noise models combined with different user simulations. We then show that policies trained in high-noise conditions perform significantly better than those trained for low-noise conditions, even when deployed in low-noise environments.

1 Introduction

For any dialogue system, a major development effort is in designing the *dialogue policy* of the system, that is, which dialogue actions (e.g. `ask(destination_city)` or `explict_confirm`) the system should perform. Machine-learning approaches to dialogue policies have been proposed by several authors, for example (Levin et al., 2000; Young, 2000; Henderson et al., 2005). These approaches are very attractive because of their potential in efficient development and automatic optimization of dialogue systems.

We will address the issue of whether policies trained for one dialogue situation can be used successfully in other dialogue situations (Paek, 2006).

For example, perhaps you have trained an optimal policy for an operating environment where the word-error rate (WER) is 5%, but you want to deploy this policy for a new application where you are not sure what the average WER is. So, you want to know how well the policy *transfers* between operating situations. Likewise, perhaps you have trained a policy on a data set of cooperative users, but you want to know how that policy will behave in contact with less co-operative users. So, you want to know how useful the policy is with different users.

These transfer issues are important because when deploying a real dialogue application we will not know these parameters exactly in advance, so we cannot train for the exact operating situation, but we want to be able to learn robust dialogue policies which are transferable to different noise/user/time-penalty situations, which we do not know about precisely before deployment.

1.1 Related work

The issue of policy transfer has been partially explored before as part of recent work on types of user simulations (Schatzmann et al., 2005). Here, the authors explore how well policies trained on different types of user simulation perform when tested with others. They train and test on three approaches to user simulation: a bigram model (Eckert et al., 1997), the Pietquin model (Pietquin, 2004), and the Levin model (Levin et al., 2000). They show that strategies learned with a "poor" user model can appear to perform well when tested with the same user model, but perform badly when tested on a "better" user model. However, the focus of (Schatzmann et al., 2005) is on the quality of the user simulation techniques themselves, rather than robustness of the learned dialogue policies. We will focus on one type of stochastic user simulation but different types of

Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue, pages 55–58,
Antwerp, September 2007.

users and on different environmental conditions.

(Frampton and Lemon, 2006) train a policy for 4-gram stochastic user simulation and test it on a 5-gram simulation, and vice-versa, showing that the learned policy works well for the 2 different simulations. However, these simulations are trained on the same dataset (Walker et al., 2001) and thus do not simulate different *types* of user or noise conditions. Similarly (Henderson et al., 2005) test and train on different segments of the COMMUNICATOR data, so the results presented there do not deal with the issue of policy transfer. (Lemon et al., 2006) show that a single policy trained on a human-machine dialogue corpus also performs well with real users of a dialogue system.

2 The experimental set-up

We experiment with a 3-slot information-seeking system, resulting in 8 binary state variables (1 for whether each slot is filled, 1 for whether each slot is confirmed, 2 for whether the last user move was "yes" or "no"), resulting in 256 distinct dialogue states. There are 5 possible system actions (e.g. implicit-confirm, greet, present-info).

We use the SHARSHA Hierarchical Reinforcement Learning algorithm of REALL (Shapiro and Langley, 2002) to learn over the policy space for obtaining 3 information slots. For all combinations of Turn Penalty, noise, and user models we train each policy on 32,000 iterations (approx. 8000 dialogues). We then test each policy (including the hand-coded policies) over 1000 dialogues in the conditions for which they were trained. Statistical significance is measured by independent samples t-tests, over 1000 test dialogues.

We use the hierarchical structure of REALL (Shapiro and Langley, 2002) programs to encode commonsense constraints on the dialogue problem, while still leaving many options for learning. The hierarchical plans encode obvious decisions such as: " never confirm already confirmed slots".

2.1 Reward function

We use a reward function which incorporates noise modelling, as in (Rieser and Lemon, 2007). For each dialogue we have, as is now commonly used:

```
reward = completionValue
         - dialogueLength*TurnPenalty
```

However, for our noise modelling, the `completionValue` of a dialogue is defined as the percentage probability that the user goal is in the actual result set that they are presented with. See (Rieser and Lemon, 2007) for full details. In our experiments Low Noise (LN) means that there is a 100% chance of confirmed slots being correct and an 80% chance of filled (but not confirmed) slots being correct. In a real application domain we will not know these probabilities exactly, but we want to be able to learn dialogue policies which are transferrable to different noise situations, which we do not know about precisely before deployment.

2.2 Simulated users

We use 2 probabilistic user simulations: "Cooperative" (C) and "Uncooperative" (U). Each simulated user produces a response to the previous system dialogue move, with a particular probablility distribution conditioned on the previous system move. For example, if the system asks for slot1 (e.g. "what type of food do you want?") the cooperative user responds to this according to the a probability distribution over dialogue acts estimated from the COMMUNICATOR corpus (Walker et al., 2001).

In contrast, the "Uncooperative" user simply has a flat probability distribution over the all the possible dialogue acts: it is just as likely to be silent as it is to supply information. This is not intended to be a particularly realistic user simulation, but it provides us with behaviour that is useful as one end of a spectrum of possible behaviours.

2.3 Baseline hand-coded policies

The hand-coded dialogue policies obey the same commonsense constraints as mentioned above but they also try to confirm all slots implicitly or explicitly (based on standard rules) and then close the dialogue, except for cases where particular dialogue length thresholds are surpassed. For example, if the current dialogue length is greater than 10 the hand-coded policy will immediately provide information.

3 Results versus hand-coded policies

In general, learning takes about 500 dialogues before a policy of confirming as many slots as possible in the shortest time is discovered. Early in the training runs the learner experiments with very short

dialogues (smaller length penalties), but usually receives less completion reward for them and so learns how to conduct the dialogue so as to trade-off between turn penalties (TP) and completion value. For example, in the High Noise, Cooperative user, turn penalty 5 case, after a policy is discovered, testing the learned policy in the same situation (but with learning and exploration turned off), the average dialogue reward is 49.94 (see figure 1, plotting average reward every 50 test dialogues, and table 1).

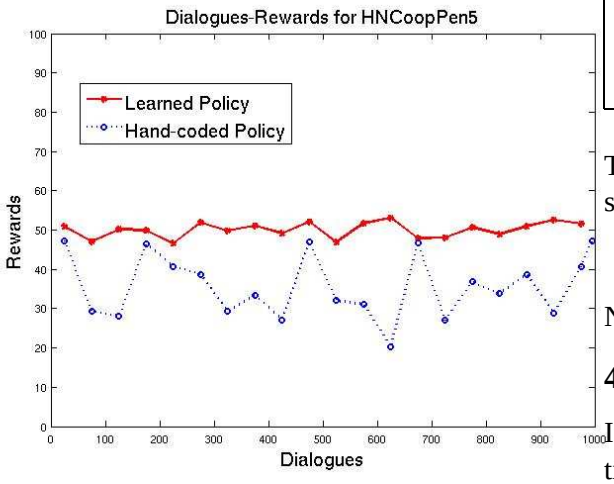

Figure 1: Testing: High noise, cooperative user, TP 5: Learned versus Hand-coded policy

Contrast this now with the performance of the hand-coded policy in the same situation (high noise, cooperative user, TP=5), over 1000 test dialogues, also shown in figure 1. The average reward for the hand-coded policy is 36.43 in these conditions, which means that the learned policy provides a relative increase in average reward of 37% in this case. This result is significant at $p < .01$.

Table 1 shows all results for the High Noise, Cooperative user case, for turn penalties (TP) ranging from 0 to 20. Here we can see that the learner is able to develop policies which are significantly better than the hand-coded policy. The exception is the TP=10 case, where the learned policy is not *significantly* better than the handcoded one ($p = .25$). For the significant results, the average relative increase in reward for learned policies is 28.4%

Considering the average dialogue lengths in each case, note that the hand-coded policy is able to complete the dialogues in, on average, fewer than 7 moves, which is less than the hand-coded length threshold (10). The learned policies, on the other hand, are able to discover their own local length/completion value trade-offs, and we see that, as expected, average dialogue length decreases as Turn Penalty increases.

	Learned Policy		Hand-coded Policy	
TP	Av. Reward	Length	Av. Reward	Length
0	85.70**	8.71	72.43	6.86
1	76.31 **	9.36	64.62	6.80
5	**49.94** **	7.18	**36.43**	6.95
10	4.16	4.05	1.77	6.89
20	-37.68 **	2.99	-63.76	6.80

Table 1: Results: Cooperative user, High Noise (**= significant at $p < .01$)

Similar results hold for the other combinations of Noise, User type, and Turn Penalty.

4 Transfer results

In the following experiments we chose to investigate the representative TP=5 case. We thus have 2 degrees of variation: user type (Cooperative/Uncooperative, C/U), and noise conditions (High/Low, H/L). Testing all combinations of these learned policies, for 1000 dialogues each, we obtained the results shown in table 5.

	Training			
Testing	C,L	C,H	U,L	U,H
C,L	**73.66**	**74.72**	54.86	54.48
C,H	**49.64**	**50.08**	21.07	25.36
U,L	23.67	27.84	**37.62**	**39.37**
U,H	09.99	14.40	**08.93**	**10.22**
Average:	39.24	*41.76*	30.62	32.36

Table 2: Transfer results for learned policies

Looking at table 2, we can see, for example, that training with a Cooperative user in Low noise (1st column) and testing with the same conditions (1st row) results in an average dialogue reward of 73.66. However, taking the same trained policy (C,L 1st column) and testing it with a Uncooperative user in High Noise conditions (row 4) results only in an average reward of 9.99. We would expect that the lead-

ing diagonal of this table should contain the highest values (i.e. that the best policy for certain conditions is the one trained on those conditions), but surprisingly, this is not the case. For example, training a C,H policy and testing it for C,L gives better results than training for C,L (and testing for C,L). This is significant at $p < .05$. This shows that a C,H policy in fact *transfers* well to C, L conditions.

Looking at the 4 policies C,L, C,H, U,L, and U,H we can see that C,H has the best transfer properties. Interestingly, C,H is the best policy for all of the testing conditions C,L, C,H, and U,H. But should we then train only in High noise conditions? Consider the following set of results (highlighted in bold font in table 5):

train C,H and test C,L $>$ train C,L and test C,L
train C,H and test C,H $>$ train C,L and test C,H
train U,H and test U,L $>$ train U,L and test U,L
train U,H and test U,H $>$ train U,L and test U,H

This indeed shows that it is better to train in High noise conditions than low noise, no matter what conditions you deploy in. These results are all significant at $p < .05$ except for the case "train C,H and test C,H $>$ train C,L and test C,H" ($p = .37$). This means that for cooperative users, training in High noise is *as good as* training in Low noise. These results show that, when training a policy for an operating environment for which you don't have much data (i.e. the developer does not yet know the noise and user characteristics) it is better to train and deploy a High noise policy, than to deploy a policy trained for Low noise conditions. Similar results show that policies trained on uncooperative users perform well when tested on cooperative users but not vice versa.

5 Conclusion

We addressed the robust of learned strategies in *transfer* to different conditions. We provided transfer results for dialogue policy learning and are the first to present results for different ASR noise models combined with different user models. We first showed that our learned policies for a range of environmental conditions (Noise, Users, Turn Penalties) significantly outperform hand-coded dialogue policies (e.g average 28% relative reward increase for cooperative users in high noise). We then compared different learned policies in terms of their transfer properties. We showed that policies trained in high-noise conditions perform significantly better than those trained for low-noise conditions, even when deployed in low-noise environments.

Acknowledgements This work is funded by the EPSRC (grant number EP/E019501/1) and by Scottish Enterprise under the Edinburgh-Stanford Link.

References

W. Eckert, E. Levin, and R. Pieraccini. 1997. User modelling for spoken dialogue system evaluation. In *Proceedings of ASRU*, pages 80–87.

Matthew Frampton and Oliver Lemon. 2006. Learning more effective dialogue strategies using limited dialogue move features. In *Proceedings of ACL*.

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

O. Lemon, K. Georgila, and J. Henderson. 2006. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *Proc. ACL/IEEE SLT*.

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

Tim Paek. 2006. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Dialogue on Dialogues*. Interspeech2006 - ICSLP Satellite Workshop.

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

V. Rieser and O. Lemon. 2007. Learning dialogue strategies for interactive database search. In *Interspeech*.

J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE ASRU Workshop*.

D. Shapiro and P. Langley. 2002. Separating skills from preference: using learning to program by reward. In *Intl. Conf. on Machine Learning.*

M. Walker, R. Passonneau, and J.Boland. 2001. Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems. In *Proc. ACL*.

Steve Young. 2000. Probabilistic methods in spoken dialogue systems. *Philosophical Transactions of the Royal Society (Series A)*, 358(1769):1389–1402.