Reward Shaping for Statistical Optimisation of Dialogue Management

Layla El Asri[1,2], Romain Laroche[1], and Olivier Pietquin[2]

[1] Orange Labs, 38-40 rue du Général Leclerc 92794 Issy-les-Moulineaux, France,
[2] SUPELEC Metz Campus, IMS-MaLIS Research group, UMI 2958 (CNRS - GeorgiaTech), 2 rue Edouard Belin 57070 Metz, France
{layla.elasri, romain.laroche}@orange.com, olivier.pietquin@supelec.fr

Abstract. This paper investigates the impact of reward shaping on a reinforcement learning-based spoken dialogue system's learning.
A diffuse reward function gives a reward after each transition between two dialogue states. A sparse function only gives a reward at the end of the dialogue. Reward shaping consists of learning a diffuse function without modifying the optimal policy compared to a sparse one.
Two reward shaping methods are applied to a corpus of dialogues evaluated with numerical performance scores. Learning with these functions is compared to the sparse case and it is shown, on simulated dialogues, that the policies learnt after reward shaping lead to higher performance.

Keywords: Spoken Dialogue Systems, Evaluation, Reinforcement Learning

1 Introduction

Dialogue management is one of the core functionalities of a Spoken Dialogue System (SDS) along with automatic speech recognition, natural language understanding, natural language generation and speech synthesis. The Dialogue Manager (DM) sequences the interaction with the user. It chooses the action to perform according to its beliefs about the current state of the dialogue. Actions the DM can perform might be: asking for a piece of information, asking the user to confirm a statement, *etc.* Hand-coding the behaviour of the DM is time consuming and results in a specific implementation difficult to transfer to other domains. Therefore, statistical learning of the DM's behaviour through Reinforcement Learning (RL) [22] has become a popular technique: the DM is modelled as a sequential decision making agent and it selects actions in order to maximise a numerical return. This return is computed from a reward function provided by the SDS designer [10]. Ideally, the reward function is to be conceived as the most succinct, robust and transferable representation of the system's task [19].

However, it is common to define this function based on SDS designer intuition and experience, not relying on any data. Only a few studies have been conducted to learn a reward function from data. Among them, Walker *et al* [24] proposed a PARAdigm for DIalogue System Evaluation (PARADISE), modelling system performance as a linear function of task completion and dialogue costs (duration of the dialogue, number of speech recognition rejections...). Walker *et al.* [23] as well as Rieser and Lemon [18] evaluated the performance of different systems using the PARADISE framework and used this evaluation as a reward function. Yet, PARADISE requires to automatically compute task completion, which is not always possible. Besides, the linear representation of system performance has been criticised for not having a strong theoretical nor experimental grounding [7]. Another methodology was proposed, which consists of learning, from examples of expert behaviour, the reward function that describes best the task being completed by that expert. This approach is known as Inverse Reinforcement Learning (IRL) [19]. It was first suggested for dialogue management by Paek and Pieraccini [15] who thought of using IRL on Human-Human dialogues to learn a reward function enabling the SDS to mimic human operators behaviour. Following this idea, Boularias *et al.* [2] learnt a reward function from dialogues collected in a Wizard-of-Oz (WOZ) setting where a human expert replaces the DM. However, it is not always possible to learn from a human expert. For example, a DM could have to choose between different speech styles and these choices can only be made statistically. Besides, it is difficult to transpose speech recognition issues to WOZ experiments. IRL has also been used to model user behaviour for dialogue simulations [4].

In previous work, we proposed two algorithms using a corpus of manually evaluated dialogues (with numerical performance scores) to compute a reward function [5]. It was shown that a reward function that predicts accurately the subjective performance for a given dialogue could be learnt, even from a corpus of small size. Sample efficiency has been an important subject of research in the field of dialogue management [11, 16]. A learning algorithm is said to be sample efficient if it can learn a near-optimal policy with only a few dialogues, meaning that it optimises data exploitation. It is costly to conduct evaluation campaigns on an SDS and most of the time, only a few number of evaluated dialogues can be collected, hence the importance of optimal data exploitation.

The reward function learnt with the methods we previously proposed gives a reward after each transition between two dialogue states while keeping the optimal policy unchanged compared to a sparser reward. We will call such a function *diffuse* in contrast with the sparse case where a reward is only received at the end of the dialogue. In this paper, it is shown, on a simulated corpus with user behaviour inferred from real dialogues, that the policy learnt with diffuse rewards entails higher performance than the one learnt with sparse rewards.

2 Reinforcement Learning for Dialogue Management

Dialogue management is cast as a sequential decision making problem, modelled by a Markov Decision Process (MDP) (S, A, T, R, γ) where S is the state space, A the action space, T the transition probabilities: $\forall\ (s = s_t, a = a_t, s' = s_{t+1})$, $T(s, a, s') = P(s' \mid s, a) \in [0, 1]$, R the reward function: $\forall (s = s_t, s' =$

$s_{t+1})$, $R(s,s') \in \mathbb{R}$ and $\gamma \in]0,1[$ a discount factor. A similar MDP without a reward function is denoted MDP$\backslash R$.

A deterministic policy π is a function mapping each state to a unique action: $\forall\ s \in S, \pi(s) = a \in A$. The immediate reward received after a transition (s_t, s_{t+1}) is $R_t = R(s_t, s_{t+1}) \in \mathbb{R}$. The cumulative reward (or return) at time t is the discounted sum of immediate rewards: $r_t = \sum_{k \geq 0} \gamma^k R_{t+k}$. Given a policy π, V^π is the state value function, the value $V^\pi(s)$ of a state s being the expected return $E[r_t \mid s_t = s, \pi]$ over all possible trajectories starting in state s_t and following π. Likewise, Q^π is the state-action value function, the value $Q^\pi(s,a)$ of a state-action couple (s,a) being $E[r_t \mid s_t = s, a_t = a, \pi]$. The dialogue manager aims to find an optimal policy: a mapping selecting actions maximising the expected return for every state. An optimal policy π^* is thus such that $\forall\ \pi, \forall\ s, V^{\pi^*}(s) \geq V^\pi(s)$. Although uniqueness of the optimal policy is not guaranteed, all optimal policies share the same state and state-action value functions noted V^* and Q^* and thus perform comparatively. In the context of this paper, time is measured in number of dialogue turns, each dialogue turn occurring in between two results of automatic speech recognition.

If many dialogue parameters are taken into account, the state space can become computationally intractable so designers usually define a summary state space instead. A summary state is an agglomerate of states with similar features. For example, for the SDS described in [8] which provides information about local restaurants, the current state can be summed up in terms of *empty*, *filled* and *confirmed* items (location, price range, type of food) instead of listing the current values of all items (*e.g.* location=city center, price range=cheap, type=Italian).

The reward function is hard to define *ex nihilo* as one should be able to numerically translate and appropriately distribute qualitative requirements. For instance, one has to decide which prevails between task completion and speech recognition rejections, when should the rewards be given during the dialogue, which numerical range, and so forth. Our approach to this problem is given in the following section which briefly presents two algorithms computing, from a corpus of manually evaluated dialogues, a diffuse reward function, defined over a summary state space $\tilde{S}$ [5]. These algorithms solve the problem introduced in Definition 1. The manual evaluations in question are based on dialogue features representative of system usability such as dialogue length, task completion,... They can be computed from user answers to a Likert-scale questionnaire. In this paper, dialogues are simulated and the scores are a linear combination of such dialogue features.

Definition 1 (Reward inference problem) *Infer a reward function from a corpus of N dialogues $(D_i)_{i \in 1..N}$ among which p dialogues have been manually evaluated with a numerical performance score $P_i \in \mathbb{R}$.*

3 Learning Rewards from Data

This section presents three different approaches to the problem issued in Definition 1. The first two algorithms infer diffuse reward functions. Details can be found in

[5]. The third one, which will serve as a baseline, gives a reward equal to system performance at the end of the dialogue.

3.1 Reward Shaping

This first algorithm is named Reward Shaping in reference to the line of research which aims to include, without modification of the optimal policy, immediate rewards as progress estimators instead of having to wait until the end of an episode to receive a reward [12]. Ng *et al.* [14] proved that the optimal policy of an MDP would not be changed by adding to the reward function a potential-based reward function U such that $U(s,s') = \gamma\Phi(s') - \Phi(s)$ and experimentally validated that, if Φ is equal to the value function, learning speed is increased. In our case, reward shaping consists of using performance scores to evaluate each state and then defining the reward associated to a given transition as the difference of potential between the arrival and the initial state.

The value function V^π is estimated according to the performance scores: the return used to estimate the value of each summary state $V^\pi(\tilde{s})$ is $\forall\ D_i$, $r_t = \gamma^{-t}P_i$. Thus, the global return r_0 is equal to P_i.

The reward function (denoted R_{RS}) is then defined as $\forall\ (\tilde{s},\tilde{s}'),\ R_{RS}(\tilde{s},\tilde{s}') = \gamma V^\pi(\tilde{s}') - V^\pi(\tilde{s}) + \delta_{\tilde{s}=\tilde{s}_0}V^\pi(\tilde{s}_0)$ with δ the Kronecker symbol ($\delta_{\tilde{s}=\tilde{s}_0} = 1$ if $\tilde{s} = \tilde{s}_0$ and 0 otherwise). In other words, the reward function is modelled as the sum of an offset $C_0 = V^\pi(\tilde{s}_0)$ and the potential-based function $U(\tilde{s},\tilde{s}') = \gamma V^\pi(\tilde{s}') - V^\pi(\tilde{s})$. With R_{RS}, the global return r_0 for a given dialogue D (lasting from turns 0 to t_f) is: $r_0 = \hat{P} = \gamma^{t_f}V^\pi(\tilde{s}_{t_f})$. Since V^π is estimated according to the returns $r_t = \gamma^{-t}P_i$, $\gamma^{t_f}V^\pi(\tilde{s}_{t_f})$ is an estimation of the performance of the dialogues ending with state $\tilde{s}_{t_f}$ and r_0 is an estimation of the performance of the system during D.

3.2 Distance Minimisation

Instead of evaluating states, distance minimisation evaluates transitions. This algorithm directly aims to cut the performance evaluations into local rewards over the transition space. The distance minimisation problem is formalised in Definition 2.

Definition 2 *Let an MDP\R. Let $\phi = [\phi_i]_{i=1,\ldots,m}$ be a vector of features over the transition space ($\forall\ i \in [1,m], \forall\ (\tilde{s},\tilde{s}')$,
$\phi_i(\tilde{s},\tilde{s}') \in [0,1]$). The immediate reward R_t following transition $(\tilde{s}_t,\tilde{s}'_t)$ is modelled as a linear sum of these features: $R_t = \sum_{i=1}^m \omega_i\phi_i(\tilde{s}_t,\tilde{s}'_t)$. Let $P = [P_i]_{i=1,\ldots,p}$ be a performance score vector such that each dialogue D_i is associated with a performance P_i, and let d_P be a distance measure between P and the return vector r_0[1]. The distance minimisation problem consists of finding the weight vector ω^* such that $\omega^* = \operatorname{argmin}_\omega d_P(\omega)$.*

[1] $(r_0)_{1\le i\le p}$, $\forall i$, $r_{0i} = \sum_{t\ge 0}\gamma^t R_t = \sum_{\tilde{s}_t,\tilde{s}'_t}\gamma^t\sum_{j=1}^m \omega_j\phi_j(\tilde{s}_t,\tilde{s}'_t)$

Here, Euclidean distance minimisation is solved. The problem issued in Definition 2 can be cast as a quadratic optimisation problem and solved with well-known direct or iterative methods (resolution details can be found in [5]). The resulting reward function is denoted R_{DM}.

3.3 Performance Scores

R_{RS} and R_{DM} are compared to the sparse reward function which gives the performance score at the end of each dialogue. This function (denoted R_{PS}) is defined as follows: $\forall\ D_i, \forall\ (\tilde{s}, \tilde{s}'),\ R_{PS}(\tilde{s}, \tilde{s}') = 0$ if $\tilde{s}' \neq \tilde{s}_f$ and $R_{PS}(\tilde{s}, \tilde{s}') = \gamma^{-t_f-1} P_i$ otherwise[2].

In the following section, the performance of the policies learnt with R_{PS}, R_{RS} and R_{DM} on the same corpus of dialogues are compared.

4 Experimental Setting

4.1 System Overview

We used the TownInfo system, based on the DIPPER architecture [1]; it provides informations about restaurants in a given city depending on three criteria: location, price range and type of food [9]. At each time step, a slot corresponding to one of these criteria can either be empty, filled or confirmed. We defined a summarised state space which counts the number of empty, filled and confirmed slots. We also defined a summary action space which does not differentiate the actions according to the position of the slot involved (for instance, AskSlot1, AskSlot2 and AskSlot3 are summarised into AskASlot). Nevertheless, to assure dialogue coherence and avoid *e.g.* asking for a slot that has already been confirmed, when an action is chosen and has to be mapped to a slot, for example, AskASlot, we first check the current value of the slots and then force this action to be mapped only to empty slots. The state and action spaces were voluntarily made simple as the main objective of this paper is to validate the diffuse rewards approach.

4.2 Dialogue Simulations

The three reward functions were applied to a corpus of 600 simulated dialogues. User was simulated according to the Bayesian method proposed in [17]. It consists of modelling user behaviour as a Bayesian network to simulate dialogues at the intention level, including grounding behaviours. The parameters of the Bayesian network were trained on the 1234 human-machine dialogues which are described in [25].

As for system policy, it was set to be uniform to collect as much information as possible for every state-action pair.

After each dialogue, a performance score was computed according to dialogue features, in a PARADISE-like manner. Once again, since our aim was to validate

[2] so that $r_0 = P_i$

the diffuse rewards approach, a simple, automatically computed scoring function was sufficient. With nbEmpty the number of empty slots, nbRight, the number of slots that were correctly filled, nbWrong, the number of incorrectly filled slots and nbTurns, the number of dialogue turns, the score was:

$$\begin{aligned}\text{score} = &- 3 \times \text{nbEmpty} + 0.25 \times \text{nbRight} \\ &- 0.75 \times \text{nbWrong} - 0.015 \times \text{nbTurns}\end{aligned} \quad (1)$$

4.3 Learning a Diffuse Reward Function

We simulated 2300 dialogues in total but we only used 600 dialogues to learn R_{RS} and R_{DM} since it is difficult, in real-life experiments to obtain as many as 2300 dialogues. The proximity between R_{RS}, R_{DM} and the simulated performance scores (see Equation 1) was assessed by Spearman's rank correlation coefficient [20]. The closer to 1 the correlation coefficient, the stronger the relationship between the corresponding rankings.

We used the remaining 1700 dialogues to measure this proximity. We drew 100 times 600 dialogues from the corpus of simulations and computed the mean correlation coefficient on these runs for both R_{RS} and R_{DM}. The mean correlation coefficient was equal to 0.81 for R_{RS} and 0.84 for R_{DM}. Here, the coefficients are high because the scoring function in equation 1 can be approximated on the state space presented in Section 4.1 as the only non-observable parameter is the number of correctly filled slots.

4.4 Learning a Near-Optimal Policy

Policies were learnt on the 600 dialogues with R_{RS}, R_{DM} and R_{PS} using Least-Squares Policy Iteration (LSPI, [6]). LSPI is an approximate policy iteration algorithm involving LSTDQ [3] which learns an approximate state-action value function for a given policy from a fixed data set. After a policy was learnt with LSPI, 200 new dialogues were generated with this policy and the dialogues were automatically evaluated according to Equation 1. We also applied LSPI to the whole corpus of 2300 dialogues and compared the three resulting policies on 200 dialogues. Our aim is to show that learning with R_{RS} and R_{DM} leads to higher performance no matter the size of the training corpus.

5 Results

Table 1 shows that though the policy learnt with R_{RS} leads to longer dialogues, it has the best evaluation. This can be explained by the fact that this policy has a better success at confirming slots than the other two. A great number of confirmed slots implies a limited risk of getting one value wrong and since filling and having the right value for each slot have a greater weight in the scoring

[3] an extension to control problems of Least-Squares Temporal Differences, LSTD [3]

Table 1. 95% confidence interval for the mean performance, mean number of dialogue turns and mean number of empty and confirmed slots on 200 dialogues simulated with the policies learnt with R_{RS}, R_{DM} and R_{PS} after 600 and 2300 dialogues.

Learning on 600 dialogues	Performance	Turns	Empty	confirmed
R_{RS}	0.13 ± 0.09	11.6	0	3
R_{DM}	0.062 ± 0.10	7.72	0	0
R_{PS}	0.007 ± 0.10	8.55	0	0.8
Learning on 2300 dialogues	Performance	Turns	Empty	confirmed
R_{RS}	0.13 ± 0.09	11.6	0	3
R_{DM}	0.08 ± 0.10	7.28	0	0.23
R_{PS}	0.04 ± 0.11	7.95	0	1.22

function (see Equation 1) than having short dialogues, the policy learnt with R_{RS} achieves better performance than the ones learnt with R_{DM} and R_{PS}. For R_{RS}, the results are the same after 600 and 2300 dialogues because the policies learnt with LSPI on these two corpora are similar.

R_{PS} gives the exact scores as rewards which makes it more accurate than R_{RS} and R_{DM} but this accuracy is counterbalanced by the fact that the rewards are only given at the end of each dialogue. The policy learnt with R_{PS} is the least competitive because it more poorly balances the trade-off between the number of confirmed slots and dialogue length than the other two policies.

6 Relation to Prior Work

Walker *et al.* [23] used performance evaluation to learn a policy for an SDS with Q-Learning [22], giving a reward equal to the evaluation at the end of each dialogue. This SDS granted a vocal access to the user's e-mail account and could summarise and read messages. Walker *et al.* showed that about hundred dialogues were sufficient to learn the best strategy between system and mixed initiative yet it was not enough for the summary strategy to achieve convergence. We showed, on a different dialogue task, that it is possible to shape a reward function based on performance evaluation in order to optimise corpus exploitation. We believe that reward shaping is a promising method for statistical dialogue management optimisation as it is often difficult to obtain corpora of great size.

Meguro *et al.* [13] designed a listening-oriented dialogue system and inferred a reward function from third-party evaluation of user satisfaction. In order to counter inter-annotators ambiguity concerning the interpretation of the Likert scale, Sugiyama *et al.* [21] introduced Preference-based Inverse Reinforcement Learning (PIRL): performance scores are used to deduce the best of two dialogues and then a reward function that classifies dialogues respecting the same order is learnt. Contrary to our reward inference algorithms, this method does not enable to use directly performance scores given by users. Indeed, each user would have to interact several times with the system for us to infer a ranking from their evaluation, otherwise a third-party annotator would be required.

7 Conclusion

This paper provided some empirical results on the issue of reward function design for spoken dialogue systems. A diffuse reward function was learnt from a corpus of evaluated dialogues. It was shown that diffuse rewards enabled to learn a policy leading to a better performance on new dialogues.

Future work will include defining a compatible active learning framework and proposing a method to optimise the conception of the summary state space. We will also compare our reward inference methods to Preference-Based Inverse Reinforcement Learning on dialogues evaluated by a third-party annotator.

Acknowledgements

The authors would like to thank Heriot-Watt University's *Interaction lab* for providing help with DIPPER. The authors would also like to thank PhD student Edouard Klein from the IMS-MaLIS research group in Supélec for providing a C implementation of LSPI and Olivier Pietquin thanks Région de Lorraine for financial support.

References

1. Bos, J., Klein, E., Lemon, O., Oka, T.: DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In: Proceedings of SIGdial Workshop on Discourse and Dialogue (2003)
2. Boularias, A., Chinaei, H.R., Chaib-draa, B.: Learning the reward model of dialogue pomdps from data. In: Proceedings of NIPS (2010)
3. Bradtke, S.J., Barto, A.G.: Linear least-squares algorithms for temporal difference learning. Machine Learning 22, 33–57 (1996)
4. Chandramohan, S., Geist, M., Lefèvre, F., Pietquin, O.: User simulation in dialogue systems using inverse reinforcement learning. In: Proceedings of Interspeech (2011)
5. El-Asri, L., Laroche, R., Pietquin, O.: Reward function learning for dialogue management. In: Proceedings of STAIRS (2012)
6. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research 4, 1107–1149 (2003)
7. Larsen, L.B.: Issues in the evaluation of spoken dialogue systems using objective and subjective measures. In: Proceedings of IEEE ASRU. pp. 209–214 (2003)
8. Lemon, O., Georgila, K., Henderson, J., Stuttle, M.: An ISU dialogue system exhibiting reinforcement learning of dialogue policies: Generic slot-filling in the talk in-car system. In: Proceedings of EACL (2006)
9. Lemon, O., Georgila, K., Henderson, J., Stuttle, M.: An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system. In: Proceedings of EACL (2006)
10. Lemon, O., Pietquin, O.: Machine learning for spoken dialogue systems. In: Proceedings of Interspeech. pp. 2685–2688 (2007)
11. Li, L., Williams, J.D., Balakrishnan, S.: Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In: Proceedings of Interspeech (2009)

12. Mataric, M.J.: Reward functions for accelerated learning. In: Proceedings of ICML. pp. 181–189 (1994)
13. Meguro, T., Higashinaka, R., Minami, Y., Dohsaka, K.: Controlling listening-oriented dialogue using partially observable markov decision processes. In: Proceedings of Coling (2010)
14. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of ICML. pp. 278–287 (1999)
15. Paek, T., Pieraccini, R.: Automating spoken dialogue management design using machine learning : An industry perspective. Speech Communication 50, 716–729 (2008)
16. Pietquin, O., Geist, M., Chandramohan, S., Frezza-Buet, H.: Sample-efficient batch reinforcement learning for dialogue management optimization. ACM Transaction on Speech and Language Processing 7(3), 1–21 (2011)
17. Pietquin, O., Rossignol, S., Ianotto, M.: Training Bayesian networks for realistic man-machine spoken dialogue simulation. In: Proceedings of IWSDS 2009 (2009)
18. Rieser, V., Lemon, O.: Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets. Computational Linguistics 37 (2011)
19. Russell, S.: Learning agents for uncertain environments (extended abstract). In: Proceedings of COLT (1998)
20. Spearman, C.: The proof and measurement of association between two things. American Journal of Psychology 15, 72–101 (1904)
21. Sugiyama, H., Meguro, T., Minami, Y.: Preference-learning based Inverse Reinforcement Learning for Dialog Control. In: Proceedings of Interspeech (2012)
22. Sutton, R.S., Barto, A.G.: Reinforcement Learning. An introduction, pp. 56–57. MIT Press (1998)
23. Walker, M.A., Fromer, J.C., Narayanan, S.: Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In: Proceedings of COLING/ACL. pp. 1345–1352 (1998)
24. Walker, M.A., Litman, D.J., Kamm, C.A., Abella, A.: PARADISE: a framework for evaluating spoken dialogue agents. In: Proceedings of EACL. pp. 271–280 (1997)
25. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Computer Speech and Language 21, 231–422 (2007)