STAIRS 2012
K. Kersting and M. Toussaint (Eds.)
© 2012 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-096-3-95

Reward Function Learning for Dialogue Management

LAYLA EL ASRI [a,b,c,1], ROMAIN LAROCHE [a] and OLIVIER PIETQUIN [b,c]
[a] *Orange Labs, Issy-les-Moulineaux, France*
[b] *UMI 2958 (CNRS - GeorgiaTech), France*
[c] *SUPELEC Metz Campus, IMS-MaLIS Research Group, France*

Abstract. This paper addresses the problem of defining, from data, a reward function in a Reinforcement Learning (RL) problem. This issue is applied to the case of Spoken Dialogue Systems (SDS), which are interfaces enabling users to interact in natural language. A new methodology which, from system evaluation, apportions rewards over the system's state space, is suggested. A corpus of dialogues is collected on-line and then evaluated by experts, assigning a numerical performance score to each dialogue according to the quality of dialogue management. The approach described in this paper infers, from these scores, a locally distributed reward function which can be used on-line. Two algorithms achieving this goal are proposed. These algorithms are tested on an SDS and it is showed that in both cases, the resulting numerical rewards are close to the performance scores and thus, that it is possible to extract relevant information from performance evaluation to optimise on-line learning.

1. Introduction

Spoken Dialogue Systems (SDS) are interfaces with which users can interact using natural language. Classically, an SDS is composed of five core components: Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), Dialogue Manager (DM), Natural Language Generation (NLG) and Text To Speech (TTS). The DM is in charge of the course of the interaction with the user: it selects system actions depending on the current state of the dialogue, which is a set of past observations and beliefs. Reinforcement Learning (RL, Sutton and Barto, 1998) has been suggested for dialogue management in order to free designers from having to entirely implement the strategy of an SDS. In this context, dialogue management is modelled as a sequential decision making problem and is then cast as a Markov Decision Process (MDP, Levin *et al.*, 1997) or a Partially Observable Markov Decision Process (POMDP, Roy *et al.*, 2000). The DM selects an action, at a given state, in an attempt at maximising the expected cumulative reward. The reward function is thereby a concise description of the task ascribed to the system. Nonetheless, there is no general framework as for the definition of this function. Most of the time, it is based on designer's experience and intuition. Paek [2006] even

[1]Corresponding Author: Layla El Asri, Orange Labs, 38-40 rue du Général Leclerc 92130 Issy-les-Moulineaux, France; E-mail: layla.elasri@orange.com.

describes the reward function as "the most hand-crafted aspect" of RL. Indeed, only a few studies have been conducted to learn the reward function from data instead of having it defined by designers.

Walker *et al.* [1997] designed a PARAdigm for DIalogue System Evaluation (PARADISE), assessing system performance in terms of the maximisation of user satisfaction along with the optimisation of dialogue costs such as dialogue duration or the number of rejections from speech recognition. Afterwards, Walker *et al.* [1998] as well as Rieser and Lemon [2011] evaluated user satisfaction according to the PARADISE framework and then used this evaluation as a reward function. However, many critics have been expressed concerning PARADISE. Among them, Larsen [2003] pointed out the fact that the suggested representation of performance as a linear function of task completion and dialogue costs had no theoretical nor experimental grounding. Besides, task completion might not always be automatically computable, which makes this approach difficult to apply to on-line learning.

Another approach aiming to learn from data the reward function of an RL-based DM is Inverse Reinforcement Learning (IRL) [Russell, 1998] which consists of learning, from examples of expert behaviour, the reward function that describes best the task being completed by that expert. Paek and Pieraccini [2008] first suggested to use IRL on Human-Human dialogues in order to learn a reward function that would enable the SDS to mimic human operators behaviour. In this spirit, Boularias *et al.* [2010] learnt a reward function for a POMDP-based SDS from human-human dialogues, in a Wizard-of-Oz (WOZ) setting, where a human expert takes the place of the DM: the expert is provided with user interaction after it was processed by speech recognition and language understanding and, given this noisy written entry, chooses system interaction. Nevertheless, WOZ experiments are expensive processes. Besides, it is not always possible to learn from a human expert. For example, a DM could have to choose between different speech styles and then, no human could assure that one speech style is better than the others: these choices can only be made statistically. Another application of IRL was proposed by Chandramohan *et al.* [2011] for user simulation. User is modelled as an MDP and IRL enables to learn the reward function followed by the user from examples of expert behaviour. The resulting simulator can adapt its strategy to modifications of system dialogue management.

This paper introduces a new methodology for learning from data a reward function for an RL-based DM. The inference of the reward function is made on the basis of a corpus of dialogues collected and then evaluated by experts, who are given simple instructions in order to limit evaluation costs. This paper presents two algorithms which translate this evaluation into a locally distributed reward function that can be used for on-line learning. From the evaluation given by the experts, these algorithms infer a repartition of rewards over the state space of the DM, which can significantly accelerate learning [Ng *et al.*, 1999]. Learning speed is paramount for SDS as it is often difficult to gather enough dialogues to learn an optimal strategy in a short period of time. The algorithms were tested on a corpus of evaluated dialogues and the resulting reward function, for both algorithms, is close to the evaluation of the dialogues. Thus, the apportionment over the whole state space is well representative of the initial evaluation.

This paper is organised as follows. Section 2 outlines the RL setting and the problem to be solved. Section 3 describes the inference algorithms and then, Section 4 compares their respective strengths and weaknesses, presenting the results of their tests on a corpus of annotated and evaluated dialogues for a given SDS. Finally, Section 5 suggests directions for future work.

2. Notations

Dialogue management is modelled as a sequential decision making problem cast as an MDP (S, A, T, R, γ) where S is the state space, A the action space, T the transition probabilities: $\forall\ (s,a,s'), T(s,a,s') = P(s' \mid s,a) \in [0,1]$, R the reward function: $\forall (s,s'), R(s,s') \in \mathbb{R}$, and $\gamma \in [0,1]$ a discount factor. This research was applied to the particular framework of Module-Variable Decision Processes (MVDP) and it implied that the reward function R should be defined over transitions and not states (see Laroche *et al.*, 2009 for more details).

A policy π is a function mapping states to actions: $\forall\ s \in S, \pi(s) = a \in A$. $R_t = R(s_t, s_{t'}) \in \mathbb{R}$ is the immediate reward received at time t, after observing transition $(s_t, s_{t'})$. The cumulative reward (or return) at time t is defined as $r_t = \sum_{t' \geq t} \gamma^{t'-t} R_{t'}$. Given a policy π, the value $V^{\pi}(s)$ of a state s is the expected return $E[r_t \mid s_t = s, \pi]$. Likewise, the value $Q^{\pi}(s,a)$ of a state-action couple (s,a) is $Q^{\pi}(s,a) = E[r_t \mid s_t = s, a_t = a, \pi]$. The aim of the DM is to find an optimal policy, which is a mapping that selects actions maximising the expected return. An optimal policy π^* is such that $\forall\ \pi, \forall\ s, V^{\pi^*}(s) \geq V^{\pi}(s)$, the corresponding state value and state-action value functions are respectively V^* and Q^*. In all that follows, time is measured in number of dialogue turns, a dialogue turn being the time elapsed between two speech recognition results.

The exact state space S of an SDS can be computationally intractable so designers usually resort to summary state spaces. Summary states are defined as groups of states sharing similar features. For instance, for a form-filling SDS, the state can be summed up as confirmed and unconfirmed items instead of reasoning upon the value of each item. Section 4.2 discusses the conception of this summary state space.

In this context, the problem to be solved is the following. A corpus of dialogues $D_1, ..., D_N$ has been collected and, among this corpus, p dialogues have been evaluated by experts. The evaluation of a dialogue D_i consists of a numerical performance score $P_i \in [-1,1]$. From this evaluation, we seek for a reward function R which will guide dialogue management towards optimal performance through on-line learning.

The following section proposes two algorithms computing such a reward function, defined over a summary state space $\tilde{S}$. These algorithms are generic in the sense that they are not based on any particular type of RL, they might be applied with Monte Carlo evaluation and control as well as temporal differences or dynamic programming [Sutton and Barto, 1998].

3. Algorithms

3.1. Reward shaping

This first approach estimates the value of each state according to the performance scores and then, uses this estimation to model the reward function as the sum of an offset $C_0 = V^{\pi}(s_0)^2$ and a potential-based function $U(s,s') = \gamma V^{\pi}(s') - V^{\pi}(s)$.

^{2}estimated as the mean of the performance scores

First, performance scores are used as returns. Let P_i be the performance score for dialogue D_i; the return at time t is defined as: $r_t = \gamma^{-t} P_i$. As it was noticed by Walker *et al.* [1997], Larsen [2003] and Laroche *et al.* [2011], the number of turns (or, equivalently, the elapsed time) is often a deciding factor in performance evaluation. Indeed, it is common to assess that, among two dialogues both leading to task completion, the best one is the shortest. Thereby, r_t is defined as $\gamma^{-t} P_i$ and not $\gamma^{t_{f_i}-t} P_i$ (t_{f_i} being the final turn of D_i). Otherwise, r_0 would be equal to $\gamma^{t_{f_i}} P_i$ and t_{f_i} would be counted twice in the return (once with the discount factor, and another time, hidden in P_i).

The second step of the algorithm is the estimation of the value of each state-action couple ($Q(s,a)\ \forall(s,a)$), from which is deduced a policy π_1. Then, $V^{\pi_1}(\tilde{s})$ is evaluated for each summary state $\tilde{s}$. The reward function is then defined as: $\forall\ (\tilde{s},\tilde{s}')$, $R(\tilde{s},\tilde{s}') = U(\tilde{s},\tilde{s}') + \delta_{\tilde{s}=\tilde{s}_0} C_0$, with δ the Kronecker symbol. Thus, the performance estimate $\hat{P}$ for each dialogue D is equal to $\gamma^{t_f} V^{\pi_1}(\tilde{s}_{t_f})$, with $\tilde{s}_{t_f}$ the final summary state of D:

$$\begin{aligned}
\hat{P} = r_0 = \sum_{t\geq 0} \gamma^t R(\tilde{s}_t,\tilde{s}_{t+1}) + C_0 &= \sum_{t=0}^{t_f-1} \gamma^t (\gamma V^{\pi_1}(\tilde{s}_{t+1}) - V^{\pi_1}(\tilde{s}_t)) + V^{\pi_1}(\tilde{s}_0) \\
&= \sum_{t=0}^{t_f-1} \gamma^{t+1} V^{\pi_1}(\tilde{s}_{t+1}) - \sum_{t=0}^{t_f-1} \gamma^t V^{\pi_1}(\tilde{s}_t) + V^{\pi_1}(\tilde{s}_0) \\
&= \sum_{t=1}^{t_f} \gamma^t V^{\pi_1}(\tilde{s}_t) - \sum_{t=0}^{t_f-1} \gamma^t V^{\pi_1}(\tilde{s}_t) + V^{\pi_1}(\tilde{s}_0) \\
&= \sum_{t=1}^{t_f-1} \gamma^t V^{\pi_1}(\tilde{s}_t) + \gamma^{t_f} V^{\pi_1}(\tilde{s}_{t_f}) - V^{\pi_1}(\tilde{s}_0) - \sum_{t=1}^{t_f-1} \gamma^t V^{\pi_1}(\tilde{s}_t) + V^{\pi_1}(\tilde{s}_0) \\
&= \gamma^{t_f} V^{\pi_1}(\tilde{s}_{t_f}) \qquad (1)
\end{aligned}$$

Likewise, the return at time t is defined as $\forall\, t > 0, r_t = \gamma^{t_f-t} V^{\pi_1}(\tilde{s}_{t_f}) - V^{\pi_1}(\tilde{s}_t)$. This reward function is then used to update the estimation of Q, from which results a new policy according to which the estimation of V is updated. This process is repeated until the value of V has converged. The entire method is described in Algorithm 1.

Seminal work by Ng *et al.* [1999] showed that, in the context of MDP-based RL, adding a potential-based function F to a reward function R_0 did not change the optimal policy. Using the optimal value function to model the potential-based function can significantly increase learning speed, which is a desired property for an on-line learning SDS and also, more generally, in batch learning, where often only a few reinforcement episodes can be gathered. Here, V is estimated according to performance scores so the resulting rewards are designed in order for the system to learn to optimise its performance. On the other hand, with our model of rewards, the return (see equation 1) only depends on the final state s_{t_f} so the corresponding optimal policy would be random. Nevertheless, this difficulty can be overcome if the summary state space is expressive enough and the final dialogue state is a good indicator of dialogue performance. An example of such a summary state space is given in Section 4.2.

As it was said previously, Algorithm 1 can be used with any type of RL. Based on the evaluation and control techniques deployed by the RL method, the state-action value function Q^{π_k} and the policy π_k are updated at each step.

Algorithm 1 Reward shaping algorithm

```
Require: the evaluated dialogues $D_1,...,D_p$ with performance scores $P_1,...,P_p$; the
  global corpus of dialogues $D_1,...,D_p,...,D_N$; a stopping criterion $\varepsilon$
  for all $D_i \in D_1,..,D_p$ do
    for all decision $d_t \in D_i$ (score $P_i$) do
      Compute the return $\forall\, t,\ r_t = \gamma^{-t} P_i$
    end for
  end for
  for all $(s,a)$ do
    Update the state-action value function: $Q^{\pi_0}(s,a) = \frac{1}{n(s,a)} \sum_{s_t=s,a_t=a} r_t$ where $n(s,a)$ is
    the number of visits to $(s,a)$.
  end for
  Update the policy: $\pi_1$
  repeat
    for all $s$ corresponding to summary state $\tilde{s}$ do
      Update the summary state value function $V^{\pi_k}(\tilde{s})$ using $\pi_k$ and $P_1,...,P_D$
    end for
    for all $D_i \in D_1,..,D_N$ do
      $R(\tilde{s},\tilde{s}') = \gamma V^{\pi_k}(\tilde{s}') - V^{\pi_k}(\tilde{s})$
      $R(\tilde{s}_0) = V^{\pi_k}(\tilde{s}_0)$
      for all $(s,a)$ do
        Update the state-action value function $Q^{\pi_k}(s,a)$ using $R$ and $\pi_k$
      end for
      Update the policy: $\pi_{k+1}$
    end for
  until $||V^{\pi_k} - V^{\pi_{k-1}}|| \leq \varepsilon$

  return R
```

The estimation of $V^{\pi_k}(\tilde{s})\ \forall\ \tilde{s}$ depends on the policy π_k but V^{π_k} is evaluated on the summary state space, which is not the space that serves as a basis for decisions, so, it cannot be computed as $\text{argmax}_a Q^{\pi_k}(s,a)$. V^{π_k} is updated in an off-policy fashion, as a weighted mean of returns [Sutton and Barto, 1998]. It is important to notice that in order to be able to compute this estimation, it is necessary for the corpus of evaluated dialogues to contain observations for each of the dialogue summary states. Generalisation of reward shaping to unknown states has recently been studied [Konidaris and Barto, 2006] and its application to this algorithm will be the subject of future work.

Next section proposes a second algorithm which differs from reward shaping in that it directly estimates from trajectories the value of transitions, without relying on policy evaluation.

3.2. Distance minimisation

Freire da Silva *et al.* [2006] introduced Inverse Reinforcement Learning with Evaluation (IRLE) which, as inverse reinforcement learning, aims to determine the reward function being optimised by an expert. In an IRLE problem, instead of having examples of expert trajectories, it is supposed that there exists an evaluator which can decide the best of two policies. The inference problem presented in this paper is close to the one described by Freire da Silva *et al.*. Indeed, a utility function for the system is deduced from the evaluation of p dialogues $D_1,...,D_p$ with performance scores $P_1,...,P_p$. The difference here with the approach of Freire da Silva *et al.* is that an optimal policy cannot be inferred as the one preferred by the evaluator, only the reward function which best fits this evaluation can be found. Besides, instead of having relative evaluations of pairs of trajectories, the evaluator provides a numerical performance score for each dialogue so, the reward function which is closest to the evaluation model can be directly computed. Distance minimisation is formalised in Definition 1.

Definition 1 *Let an MDP$\backslash$R $[S,A,T,\gamma]$. Let $\phi = [\phi_i]_{i=1,...,m}$ be a vector of features over the transition space ($\forall\ i \in [1,m], \forall\ (s,s'), \phi_i(s,s') \in [0,1]$), $P = [P_i]_{i=1,...,p}$ be a performance score vector such that each dialogue D_i, $i = 1,..,p$ is associated with a performance P_i, and d_P be a distance measure between P and the reward vector $R = w^T\phi$. The distance minimisation problem consists of finding w^* such that $w^* = \operatorname{argmin}_w d_P(w)$.*

As in the previous section, for tractability, rewards are defined over a summary state space $\tilde{S}$. For a dialogue D, the return $r(D)$ is defined as a function of the features $\phi_1,...\phi_m$.

$$\begin{aligned} r(D) &= \sum_{t\geq 0}\gamma^t R_t = \sum_{\tilde{s}_t,\tilde{s}'_t}\gamma^t\sum_{i=1}^{m} w_i\phi_i(\tilde{s}_t,\tilde{s}'_t) \\ &= \sum_{i=1}^{m} w_i\sum_{\tilde{s}_t,\tilde{s}'_t}\gamma^t\phi_i(\tilde{s}_t,\tilde{s}'_t) = w^{\mathrm{T}}\Phi(D) \qquad (2) \\ &\text{with } \Phi(D) = (\Phi_1(D)...\Phi_m(D))^{\mathrm{T}} \text{ and } \Phi_i(D) = \sum_{\tilde{s}_t,\tilde{s}'_t}\gamma^t\phi_i(\tilde{s}_t,\tilde{s}'_t) \end{aligned}$$

In what follows, Euclidean distance minimisation is solved, thereby the reward function being looked for is the one which is closest to the evaluation from a purely numerical point of view. Section 5 discusses the choice of the distance measure. The optimisation problem is the following:

$$\text{minimise } d_P(w) = \sum_{l=1}^{p}(r(D_l) - P(D_l))^2 = \sum_{l=1}^{p}(w^{\mathrm{T}}\Phi(D_l) - P(D_l))^2$$

In a matrix form:

$$\text{minimise } \frac{1}{2}w^{\mathrm{T}}[2\sum_{l=1}^{p}\Phi(D_l)\Phi^{\mathrm{T}}(D_l)]w - w^{\mathrm{T}}[2\sum_{l=1}^{p}P(D_l)\Phi(D_l)] = w^{\mathrm{T}}[\frac{1}{2}Mw - b]$$

$$\text{with } M = 2\sum_{l=1}^{p}\Phi^{\mathrm{T}}(D_l)\Phi(D_l) \text{ and } b = 2\sum_{l=1}^{p}P(D_l)\Phi(D_l) \qquad (3)$$

3.2.1. Resolution

Matrix M is symmetric. When M is positive and definite, the optimisation problem described in equation 3 has a unique solution and it is tantamount to solving the equation $Mw = b$.

M is positive. Let $x \in \mathbb{R}^m : x^{\mathrm{T}}Mx = \sum_{i=1}^{m} x_i \sum_{j=1}^{m} x_j m_{i,j} = \sum_{D_l} 2\left(\sum_{i=1}^{m} x_i \Phi_i(D_l)\right)^2 \geq 0$. Under certain conditions, M is definite. Let $x \in \mathbb{R}^m$, according to the precedent derivation: $x^{\mathrm{T}}Mx = 0 \Leftrightarrow \forall\ D_l, \sum_i x_i \Phi_i(D_l) = 0$. Put down $L = \left(\sum_{t_j \in D_l} \gamma^{t_j} \phi_i(\tilde{s}_{t_j}, \tilde{s}'_{t_j})\right)_{l,i}$. L is a rectangular matrix of size: the number of dialogues (p) $\times$ the number of transitions (m). M is definite if and only if m dialogues can be selected such that, on this new corpus, all the transitions have been observed at least once and one cannot find a pair of transitions which would be systematically correlated in time. Indeed, if this corpus can be found, let L' be the matrix formed with the lines of M corresponding to these m dialogues. According to what precedes: $\forall\ x \in \mathbb{R}^m$, $L'x = 0$. L' was chosen so that it would have a rank equal to m so, its kernel is the empty set. Therefore, $L'x = 0 \Rightarrow x = 0$ and M is definite. The reverse implication can easily be proved by contradiction.

When M is definite, $Mw = b$ admits a unique solution which can be computed using either a direct (Cholesky decomposition...) or an iterative (conjugate gradient...) method. When M is not definite, the solution to the minimisation problem described in equation 3 is not unique. The problem can be solved using Tikhonov regularisation and then, it consists of searching for w^* that minimises $\|Mw - b\|^2 + \|\delta w\|^2$. The parameter δ can be fixed with the L-curve method or cross-validation.

4. Tests

4.1. SDS architecture

Both reward inference approaches were tested on an SDS (System 3 in Laroche *et al.*, 2011) that took part to the CLASSiC European project[3] evaluation. This system was designed as an automaton. Some nodes of the automaton include a *module*, which is a

[3]Computational Learning in Adaptive Systems for Spoken Conversation, http://www.classic-project.org/

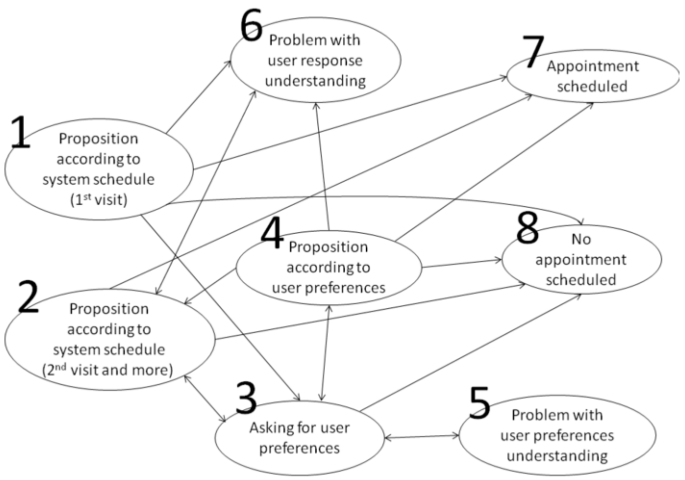
6
Problem with user response understanding
7
Appointment scheduled
1
Proposition according to system schedule (1st visit)
4
Proposition according to user preferences
8
No appointment scheduled
2
Proposition according to system schedule (2nd visit and more)
3
Asking for user preferences
5
Problem with user preferences understanding

Figure 1. A schematic version of the SDS used for experimentations. Each of the 8 modules is identified by a number.

decision point: a module selects an action according to the current state of the dialogue. This system enables clients to schedule an appointment with a technician whenever they have troubles with their landlines. Each of its 8 modules has a state space of dimension 1 so one module corresponds to one state. At each state, the system can execute three different actions, consisting of three alternatives of speech style: neutral, calm or dynamic [Janarthanam *et al.*, 2011]. A schematic display of this system is given in Figure 1, where a node of the graph stands for a module and an arrow from one module to another indicates a possible transition between these modules during a dialogue (possibly after 0 or several dialogue turns).

A dialogue is a succession of phases. The classical course of a dialogue phase starts with a declaration from the system, followed by an answer from the user depending on which the next phase is decided. For example: the system suggests the user to propose a date for an appointment (module 3), the user answers with a date, which leads to the next phase starting with the system telling the user whether this date is available (module 4) or not (module 2).

CLASSiC System 3 was evaluated on the basis of 740 dialogues. For each dialogue, the performance of the system was deduced from the overall rating (between -1 and 1) given by the user who filled a PARADISE-like questionnaire after interacting with the system [Bretier *et al.*, 2010].

4.2. Summary state space

The summary state space was computed depending on the following features: the current phase or information state [Larsson and Traum, 2000] (`phase`), the number of turns (`#turns`), the number of Automatic Speech Recognition (ASR) rejections (`#ASR rejections`), the number of user time outs (`#time out`) and the ASR Score for the current user interaction (`ASRS`). Following Rieser and Lemon [Rieser and Lemon, 2011], feature discretisation [Fayyad and Irani, 1993] and correlation-based selection [Hall, 2000] was used in order to restrict the summary state space of a decision point to the most relevant set of features. Rieser and Lemon built their SDS from the dialogues recorded with a wizard being in charge of dialogue management. They selected, at each decision point, the set of features which was most representative of the decisions made by this wizard. Contrary to Rieser and Lemon, the features here were not retained according to their relevance for decision making: they were the one that, at each decision point, best explained the difference between the estimated performance expectations from this point. This choice was motivated by the fact that, here, the state space should be adapted to the evaluation and not the previous decisions which were made according to a possibly erroneous reward function. The resulting summary state space was composed of 9 states: each module was associated to its phase and module 8 was associated to two summary states: `phase` = *No appointment scheduled* and `#turns` < 12 or ≥ 13.

For distance minimisation, there were as many transition features ϕ_i as transitions τ_i: ϕ_i was equal to 1 at τ_i and 0 at the rest of the transition space.

4.3. Results

Both reward shaping and distance minimisation were applied to the corpus of 740 evaluated calls. All the transitions between summary states had been observed at least once but distance minimisation had to resort to Tikhonov regularisation because matrix M was not invertible. Indeed, as one can see on Figure 1, module 5 is always followed by module 3. So, transition $5 \mapsto 3$ always comes after transition $3 \mapsto 5$, which causes column dependencies in M. Other dependencies were observed which were not as direct as this one. The parameter of the regularisation was determined using the L-curve method.

Preliminary tests on the summary state space described in Section 4.2 showed that, when a dialogue did not end with module 7 or 8, in both cases, systematically, the system did not compute an appropriate reward. Indeed, when the user hangs up, the system cannot take into account task completion. Yet, the latter has been proved to be of crucial importance for users who evaluated this system [Laroche *et al.*, 2011] and also, in general, for SDS users [Larsen, 2003, Walker *et al.*, 1997]. Consequently, the performance of the dialogues which ended because the user hung up was systematically ill-evaluated by both inference algorithms. In order to overcome this drawback, a *hang up* state was added to the summary state space.

The results were compared based on the average Manhattan distance between the performance scores and the returns. $r_0(D) = \sum_{t=0}^{t_f} \gamma^t R_t$ being the return for dialogue D, the average Manhattan distance is: $\frac{1}{p} \sum_{i=1}^{p} |r_0(D_i) - P_i|$. The corpus was separated into a training set of 540 dialogues and a test set of 200 dialogues. The average Manhattan distance was equal to 0.31 for reward shaping whereas it was equal to 0.29 for distance minimisation. The algorithms were also tested with another performance metric which was defined as: $2 \times$ task completion $- 0.03 \times$ #time out $- 0.05 \times$ #ASR rejection $- 0.01 \times$ #turns. With this metric, the average Manhattan distance was equal to 0.03 for reward shaping and 0.07 for distance minimisation. Since the performance was completely determined by the same dialogue features as the ones of the state space, it was easier for both algorithms to deduce an appropriate reward function.

Figure 2 displays the returns computed by reward shaping and distance minimisation with the overall ratings given by users. Globally, the returns inferred by both algorithms are coherent with the performance scores. However, Figure 2 shows that some dialogues are largely overrated by both inferred reward functions. For example, the eighth dialogue (red points in Figure 2) induced a positive return while user evaluation on this dialogue was highly negative. This phenomenon is due to the fact that Module 7 is not completely representative of task completion. Indeed, when users accepted an appointment which was not the one they had planned, it was considered that the task had not been achieved so users gave poor ratings to these kinds of dialogues. Therefore, task completion was not fully observable by the system and the latter tended to overrate dialogues that ended with module 7, which was, most of the time, synonymous with task completion and high user rating. This also explains why dialogues ending with task completion seem underrated (blue points): the value of Module 7 included both successful dialogues and unsatisfactorily booked appointments.

5. Discussion

The algorithms which have been described represent different approaches. Indeed, reward shaping estimates the value of states and then deduces the value of transitions whereas distance minimisation directly estimates the value of a transition according to the expected return after observing this transition. In the previous experiment, both in-

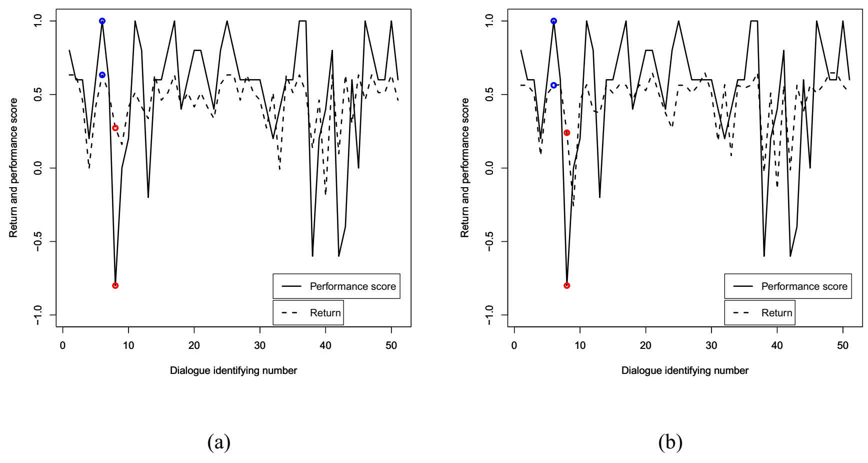

Figure 2. Comparison of performance evaluation with the return computed by reward shaping (a) and the one computed by distance minimisation (b) on 50 dialogues.

ference algorithms computed a reward function which was not sparse, *i.e.* such that the system seldom receives a reward equal to 0 after a given transition. Yet, the greatest rewards are assigned when a final state is reached and intra-dialogue transitions are often negatively or only slightly positively rewarded. Thus, the trap consisting of aiming for a sub-task should be avoided. A policy will be learnt with these reward functions and it will be analysed in future work.

An advantage of distance minimisation is that it points out system idiosyncrasies (as dependencies in matrix M), which might enable to modify system architecture or at least enhance one's comprehension of its behaviour. On the other hand, reward shaping does not make any assumption about the shape of the reward function whereas distance minimisation requires to define R as a linear function of features over the transition space. The choice of these features is strongly related to the conception of $\tilde{S}$ and will be also discussed in future work.

Furthermore, a *hang up* state was added to the summary state space. If the dialogue is modelled in such a way that, when the user hangs up, the system has failed achieving the task, then the *hang up* state can be estimated with no ambiguity. Nevertheless, this might not always be the case. Paek and Pieraccini [2008] take the example of an SDS dedicated to airline reservation. Task completion depends on the aim of the user, which might be to make a reservation or just gather information about prices. If the users hang up, the system cannot know if it is because the users have had enough information or because they give up trying to get it.

One central issue for both algorithms is the definition of the summary state space. Instead of defining the reward function as a linear function of dialogue features like Walker *et al*. did, dialogue features are included in the summary state space and the reward function is based on the evaluation of each summary state. $\tilde{S}$ must thus be coherent with system performance evaluation. Therefore, future work will consist of a better exploitation of performance scores in order to automatically compute the summary state space which best enables to distinguish performance scores. The inference algorithms will then optimise data exploitation, deducing a reward function from both summary state space learning and numerical reward learning.

The choice of the distance measure for the distance minimisation algorithm will also be the subject of future work. One can argue indeed that the Euclidean distance might not be the most appropriate choice if the aim is to imitate efficiently performance evaluation. For instance, although system learning will eventually be based on the numerical values of rewards, it might be preferable to infer a reward function which preserves the ranking of the dialogues established by the evaluation. In such a case, the distance measure would be based more on the behaviour of the functions than on their values. The same remark can be applied to the choice of the Manhattan distance to compare results from both algorithms: preserving the order of the scores might prevail over numerical proximity.

Finally, an evaluation framework which focuses on dialogue management instead of estimating general system usability will be designed. Indeed, although dialogue management is strongly related to system usability, it is not completely responsible for it. For instance, questions related to ergonomics should be avoided [Hajdinjak and Mihelic, 2007].

6. Conclusion

This paper proposed two algorithms which learn, from a corpus of dialogues evaluated by experts, a reward function for a Reinforcement Learning-based Dialogue Manager. Experts are asked, in a simple way, to assess system performance on a set of dialogues. A locally distributed reward function is then deduced from these scores. These algorithms were tested on a corpus of 740 dialogues evaluated by users and it was showed that the inferred rewards were close to the performance scores.

Future work will consist of developing both inferring approaches and a special attention will be paid to the definition of the dialogue system's summary state space. The first part of the inference process, which is dialogue evaluation, will also be tackled: an evaluation framework more precisely designed for dialogue management evaluation will be proposed.

References

Abdeslam Boularias, Hamid R. Chinaei, and Brahim Chaib-draa. Learning the reward model of dialogue pomdps from data. In *Twenty-Fourth Annual Conference on Neural Information Processing Systems*, 2010.

Philippe Bretier, Romain Laroche, and Ghislain Putois. D5.3.4: Industrial self-help system ("system 3") adapted to final architecture. report d5.3.4, classic project. Technical report, 2010.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefvre, and Olivier Pietquin. User simulation in dialogue systems using inverse reinforcement learning. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association*, 2011.

Usama M. Fayyad and Keki B. Irani. *Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning*, volume 2, pages 1022–1027. Morgan Kaufmann, 1993.

Valdinei Freire da Silva, Anna Helena Reali Costa, and Pedro Lima. Inverse reinforcement learning with evaluation. In *IEEE International Conference on Robotics and Automation*, 2006.

Melita Hajdinjak and France Mihelic. A dialogue-management evaluation study. *Journal of Computing and Information Technology*, 2007.

Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *17th International Conference on Machine Learning*, pages 359–366. Morgan Kaufmann, 2000.

Srinivasan Janarthanam, Oliver Lemon, Romain Laroche, and Ghislain Putois. D4.5: Testing learned nlg and tts policies with real users, in self-help and appointment scheduling systems. report d4.5, classic project. Technical report, 2011.

George Konidaris and Andrew Barto. Autonomous shaping : Learning to predict reward form novel states. In *26th International Conference on Machine Learning*, 2006.

Romain Laroche, Ghislain Putois, Philippe Bretier, Martin Aranguren, Julia Velkovska, Helen Hastie, Simon Keizer, Kai Yu, Filip Jurcicek, Oliver Lemon, and Steve Young. D6.4: Final evaluation of classic towninfo and appointment scheduling systems, report d6.4. classic project. Technical report, 2011.

Romain Laroche, Ghislain Putois, Philippe Bretier, and Bernadette Bouchon-Meunier. Hybridisation of expertise and reinforcement learning in dialogue systems. In *Interspeech, special session: Machine Learning for Adaptivity in Spoken Dialogue, Brighton (United Kingdom)*, 2009.

Lars Bo Larsen. Issues in the evaluation of spoken dialogue systems using objective and subjective measures. In *IEEE Workshop on Automatic Speech Recognition and Understanding ASRU'03*, pages 209–214, 2003.

S. Larsson and D. Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 6:323–340, 2000.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. Learning dialogue strategies within the markov decision process framework. In *IEEE Workshop on Automatic Speech Recognition and Understanding, Santa Barbara, California*, 1997.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999.

Tim Paek. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Dialog-on-Dialog Workshop, Interspeech, Pittsburgh, PA*, 2006.

Tim Paek and Roberto Pieraccini. Automating spoken dialogue management design using machine learning : An industry perspective. *Speech Communication*, 50:716–729, 2008.

Verena Rieser and Oliver Lemon. Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets. *Computational Linguistics*, 37, 2011.

Nicolas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialogue management using probabilistic reasoning. In *38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, 2000.

Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, 1998.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning. An introduction*, pages 56–57. MIT Press, 1998.

Marilyn A. Walker, Jeanne C. Fromer, and Shrikanth Narayanan. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *36th Annual Meeting of the Association of Computational Linguistics, COLING/ACL 98*, pages 1345–1352, 1998.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. Paradise: a framework for evaluating spoken dialogue agents. pages 271–280, 1997.