Will my Spoken Dialogue System be a Slow Learner ?

Layla El Asri
Orange Labs / UMI 2958 (IMS-MaLIS)
Issy-les-Moulineaux (France) / Metz (France)
layla.elasri@orange.com

Romain Laroche
Orange Labs
Issy-les-Moulineaux (France)
romain.laroche@orange.com

Abstract

This paper presents a practical methodology for the integration of reinforcement learning during the design of a Spoken Dialogue System (SDS). It proposes a method that enables SDS designers to know, in advance, the number of dialogues that their system will need in order to learn the value of each state-action couple. We ask the designer to provide a user model in a simple way. Then, we run simulations with this model and we compute confidence intervals for the mean of the expected return of the state-action couples.

1 Introduction

The Dialogue Manager (DM) of a Spoken Dialogue System (SDS) selects actions according to its current beliefs concerning the state of the dialogue. Reinforcement Learning (RL) has been more and more used for the optimisation of dialogue management, freeing designers from having to fully implement the strategy of the DM.

A framework known as Module-Variable Decision Process (MVDP) was proposed by Laroche et al. (2009) who integrated RL into an automaton-based DM. This led to the deployment of the first commercial SDS implementing RL (Putois et al., 2010).

Our work intends to continue this effort in bridging the gap between research advances on RL-based SDS and industrial release. One important issue concerning the design of an RL-based SDS is that it is difficult to evaluate the number of training dialogues that will be necessary for the system to learn an optimal behaviour. The underlying mathematical problem is the estimation of the training sample size needed by the RL algorithm for convergence. Yet, designers are often not experts in RL. Therefore, this paper presents a simple methodology for evaluating the necessary sample size for an RL algorithm embedded into an SDS. This methodology does not require any RL expertise from designers. The latter are asked to provide a model of user behaviour in a simple way. According to this model, numerous simulations are run and the sample size for each module-state-action triple of the DM is estimated. This methodology was tested on an SDS designed during the CLASSiC European project[1] (Laroche et al., 2011) and we show that these computations are robust to varying models of user behaviour.

2 Dialogue Management as a Module-Variable Decision Process

Module-Variable Decision Processes (MVDP) factorise learning into modules, each module having its own state and action spaces. Formally, an MVDP is a tuple (M, V_M, A_M, T) where M is the module space, V_M is the space of *local contexts*, for each module m, $V_m \subset V_M$ is the set of variables which are relevant for m's decision making. $A_m \subset A_M$ is the set of possible actions, an action beeing a transition in the automaton. $T \subset \mathbb{R}$ is the time scale. In the following, time is measured in number of dialogue turns, a turn being the time elapsed between two ASR results.

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

2.1 The Compliance Based Reinforcement Learning Algorithm

The *Compliance-Based Reinforcement Learning* algorithm (CBRL, Laroche et al., 2009) is an adaptation of the Monte Carlo algorithm to online off-policy learning. Each evaluation phase in the Monte Carlo procedure requires numerous new episodes. CBRL enables to accelerate this process by adjusting the current policy not after a set of many new episodes but right after each episode and using all the previous episodes to evaluate the policy. Each dialogue is modelled as a sequence of decisions $d_t = (m_t, s_t, a_t, t)$ where m_t is the module encountered at time t, s_t is the current local context of m_t and a_t is the action chosen by m_t. Each decision d_t leads to an immediate reward R_t. With γ a discount factor, the return for a decision d_t is $r_t = \sum_{t_i=t}^{t_f} \gamma^{t_i-t} R_{t_i}$, t_f being the final turn of the dialogue. For a given module m, the value of any state-action couple (s,a) is the expected return starting from (s,a) and then choosing actions according to π, the policy of the system: $Q^{\pi}_m(s,a) = E[r_t \mid m_t = m, s_t = s, a_t = a, \pi]$. π is the set of all the policies of the modules: $\pi = \{\pi_{m_1}, ..., \pi_{m_{|M|}}\}$. After a dialogue is taken under policy π, the value of any triple (m,s,a) is updated as in Equation 1.

$$Q^{\pi}_m(s,a) = \frac{\sum\limits_{\Theta_m(s,a)} \omega_t r_t}{\Omega_m(s,a)} \tag{1}$$

$$\text{where } \Omega_m(s,a) = \sum_{\Theta_m(s,a)} \omega_t,$$

$$\text{and } \Theta_m(s,a) = \{d_t\}_{m_t=m; s_t=s; a_t=a} \tag{2}$$

For any module m, the algorithm evaluates the value of each couple (s,a) according to all the decisions in which this tuple has been involved from the beginning of learning (the set of decisions $\Theta_m(s,a)$). After each evaluation of the Q-function, the policy π is updated following an exploratory strategy based on the *Upper Confidence Bound 1 - Tuned* approach (Auer et al., 2002). The weights ω_t in Equation 1 are there to take into account the fact that π is evaluated according to all the rewards observed since the beginning of learning, rewards that were obtained following other policies. A local compliance $c_\pi(d_t)$ is associated with each decision d_t: it is the expected regret induced by a_t not being the optimal action according to the system's current policy π, $c_\pi(d_t) = Q^{\pi}_{m_t}(s_t, a_t) - max_{a \in A_{m_t}} Q^{\pi}_{m_t}(s_t, a)$. The global compliance with π of the decisions following d_t is a discounted sum of the local compliances. The weight w_t is then an increasing function of the global compliance.

3 Problem Resolution

3.1 Approach

The problem to be solved is the following. Let an MVDP (M, V_M, A_M, T). For each triple (m,s,a), we want to compute the error made on the estimate $Q_m(s,a)$ of $E[r \mid m,s,a]$ according to the number of observations $\Theta_m(s,a)$. Let $r_1, ..., r_{|\Theta_m(s,a)|}$ be the returns corresponding to the decisions in $\Theta_m(s,a)$ and $\sigma_{m(s,a)}$ the variance of these returns. We build a confidence interval for $E[r \mid m,s,a]$, centered in the estimate $Q_m(s,a)$ from user simulations with a bi-gram model specified by the designer.

3.2 User Simulations

User simulation has been an active line of research as it is often costly to gather real data (Scheffler and Young, 2002; Georgila et al., 2006; Yang and Heeman, 2007; Pietquin and Hastie, 2010). Task-oriented systems such as commercial ones aim to respond to a specific need. They are often conceived as slot-filling systems (Raux et al., 2003; Chandramohan et al., 2011). The dialogue is relatively well-guided by the system so there is no need to take into account complex conversational groundings to simulate user behaviour. Therefore, we choose here to ask the designer to provide a bi-gram model (Eckert et al., 1997): a probability distribution of user behaviour only conditioned on the latest system action. For each possible response, the designer provides a lower and an upper bound for its probability of occurring. Eckert et al. (1997) showed that slight modifications of user behaviour in the bi-gram model did not entail great differences of system performance. We support this claim in Section 4 where we show that the confidence intervals computation is robust to varying user behaviour.

3.3 Confidence Intervals

According to the Lyapunov central limit theorem, $Q_m(s,a)$ converges in law to the normal distribution of mean $E[Q_m(s,a)] = E[r \mid m,s,a]$ and variance $\mathtt{var}(Q_m(s,a)) = \frac{\sum_{\Theta_m(s,a)} w_k^2}{\Omega_m^2(s,a)} \sigma_m^2(s,a)$. However, since $\sigma_m^2(s,a)$ is unknown and the observations are not necessarily distributed according to a normal law, we can only rely on an asymptotic result according to which, for a sufficiently large number of samples, the previous convergence result holds with the unbiased estimate of the returns variance $\tilde{\sigma}_m(s,a)$. A confidence interval of probability $1-\alpha$ for $E[r \mid m,s,a]$ is then:

$$[Q_m(s,a) - \epsilon_{m,s,a}, Q_m(s,a) + \epsilon_{m,s,a}] \qquad (3)$$

We note $u_\alpha = \Phi_{N(0,1)}^{-1}(1-\frac{\alpha}{2})$, with $\Phi_{N(0,1)}$ the cumulative distribution function of $N(0,1)$:

$$\epsilon_{m,s,a} = \frac{\sum\limits_{\Theta_m(s,a)} \omega_k^2}{\Omega_m(s,a)} \tilde{\sigma}_m(s,a) u_\alpha \qquad (4)$$

In the non-weighted case, the previous asymptotic result is generally considered to hold for a number of samples greater than 30. We thus consider the confidence intervals to be valid for $\overline{\Omega}_m(s,a) = \frac{\Omega_m^2(s,a)}{\sum_{\Theta_m(s,a)} \omega_k^2} > 30$.

3.4 β-Convergence Definition

A confidence interval can be computed for each (m,s,a) triple of the system. From this computation, we deduce the number of dialogues necessary for convergence *i.e.* for the width of the confidence interval to be under a given threshold. The confidence interval radius of a triple (m,s,a) depends on the variance of observed returns (see equation 4) so we define the normalised confidence interval radius:

$$\bar{\epsilon}_{m,s,a} = \frac{\epsilon_{m,s,a}}{\tilde{\sigma}_m(s,a)} = \frac{u_\alpha}{\sqrt{\overline{\Omega}_m(s,a) - 1}} \qquad (5)$$

We will consider that a triple (m,s,a) will have β-converged once the normalised confidence interval radius will have come under β.

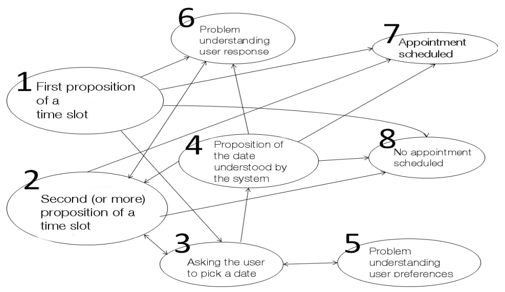
6 Problem understanding user response
7 Appointment scheduled
1 First proposition of a time slot
4 Proposition of the date understood by the system
8 No appointment scheduled
2 Second (or more) proposition of a time slot
3 Asking the user to pick a date
5 Problem understanding user preferences

Figure 1: A schematic view of the system.

4 Experiments

4.1 System Description

The negotiation strategy of the system is hard-coded (see Figure 1). The system starts each dialogue proposing to the user its first availability (module 1). Then, if the user rejects the proposition, the system asks them to give their first availability (module 3). If the first two steps have not resulted in success, the system proposes its next availabilities (module 2) until an appointment is booked (module 7) or the system has no more propositions to make (module 8). When a user proposes a date, the system asks for a confirmation through module 4. Two error-repair modules (modules 6 and 5) notify the user that they have not been understood or heard (in case of a time out). More details can be found in (Laroche et al., 2011). Each module has to choose between three actions: uttering with a calm (action 1), neutral (action 2) or dynamic (action 3) tone. In our experiments, user simulation was modelled so that the first two alternatives were artificially disadvantaged: the number of failures was slightly increased whenever one of them was chosen. We modelled here the fact that users would always prefer the dynamic intonation.

We ran 2000 simulations, one simulation consisting of a complete dialogue ending with an update of the state-action value function for each of the system's modules. The following results are averages on 100 runs.

We set the hanging-up rate to 10%. α was set to 0.05 and β to 0.1. In the following section, we use the notation (i,j,k) to refer to (m_i, s_j, a_k).[2]

[2] s_j is always equal to 1 because the local contexts space is equal to the module space

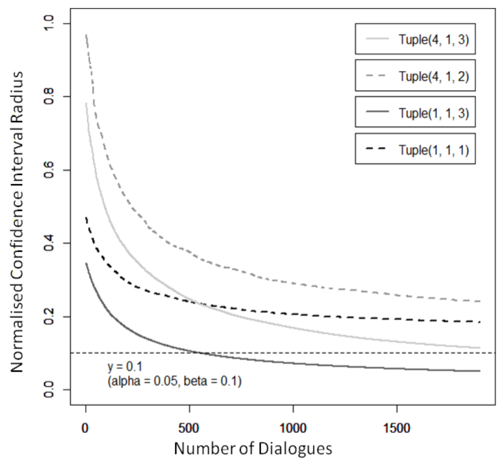

Figure 2: Evolution of $\bar{\epsilon}_{m,s,a}$ for triples (1, 1, 1), (1, 1, 3), (4, 1, 2) and (4, 1, 3) according to the total number of dialogues. Users prefer action 3.

4.2 Results

By the end of our experiments, modules 4, 5 and 8 had not $\beta_{0.1}$-converged. Module 5 was not likely to be visited quite often according to our specification of user behaviour. The same happened for module 4, only accessible from module 3 (see Figure 1), which was not itself often visited. Module 1 is, with module 8, a starting module of the system. At the beginning of a dialogue, module 1 had a 95% probability of being visited whereas this probability was of 5% for module 8 (this only happened when all available appointments had already been booked). Therefore, module 1 was visited once during almost every dialogue. We will now focus on modules 1 and 4 for clarity of presentation.

We can conclude from Figure 2 that triple (1, 1, 3) $\beta_{0.1}$-converged after about 640 dialogues, corresponding to about 425 visits whereas neither triple (1, 1, 1) nor (4, 1, 2) nor (4, 1, 3) $\beta_{0.1}$-converged, even after 2000 dialogues. Indeed, these triples did not receive enough visits during the simulations. Triple (1, 1, 3) $\beta_{0.1}$-converged whereas (1, 1, 1) did not because, at one point, the growth of the number of visits to (1, 1, 1) slowed down as module 1 favoured action 3 and reduced its exploration of other actions. The fact is that the RL algorithm did not need such a precise estimation for (1, 1, 1) to understand action 1 (the neutral tone) was suboptimal.

The variance over the 100 runs of the final estimation of $\bar{\epsilon}_{m,s,a}$ was below 0.01. For all triples of the system, the variance was very low after about 500 dialogues only (from 10^{-5} to 0.02). This means that the approximate user behaviour, defined with probability windows, only had a limited impact on the reliability of the computed confidence intervals. The probability windows used in the experiments were narrow (of an average size of 10%) so user behaviour did not change drastically from a run to another. With a behaviour much more erratic (larger probability windows), the variance over 10 runs was higher but did not exceed 0.02.

5 Related Work

Suendermann et al. (2010) tackled the issue of reducing the risk induced by on-line learning for commercial SDS with contender-based dialogue management. Our study relates to this work but within the more complex learning structure of RL.

Closer to our study, Tetreault et al. (2007) compared confidence intervals for the expected return for different MDPs, all modelling the same SDS but with a different state space. They showed how the intervals bounds as well as the expected cumulative returns estimations could be used in order to select an appropriate state space. More recently, Daubigney et al. (2011) as well as Gasic et al. (2011) developed an efficient exploration strategy for an MDP-based DM based on the uncertainties on the expected returns estimations. The difference between these two approaches and ours is that they compute the confidence intervals for a known policy whereas we compute the expected confidence intervals for an unknown policy that will be learnt on-line.

6 Conclusion

To help the development of SDS embedding on-line RL, we have designed and implemented an algorithm which computes the normalised confidence interval radius for the value of a state-action couple. We have illustrated this algorithm on an appointment scheduling SDS. We believe our method can be transferred to any system implementing an RL episodic task, as long as the environment can be simulated.

References

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. 2011. User simulation in dialogue systems using inverse reinforcement learning. In *Proceedings of Interspeech*.

Lucie Daubigney, Milica Gasic, Senthilkumar Chandramohan, Matthieu Geist, Olivier Pietquin, and Steve Young. 2011. Uncertainty management for on-line optimisation of a pomdp-based large-scale spoken dialogue system. In *Proceedings of Interspeech*, pages 1301–1304.

Wieland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *Proceedings of IEEE ASRU*, pages 80–87.

Milica Gasic, Filip Jurcicek, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Proceedings of IEEE ASRU*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Proceedings of Interspeech*.

Romain Laroche, Ghislain Putois, Philippe Bretier, and Bernadette Bouchon-Meunier. 2009. Hybridisation of expertise and reinforcement learning in dialogue systems. In *Proceedings of Interspeech*.

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

Olivier Pietquin and Helen Hastie. 2010. Metrics for the evaluation of user simulations. Technical Report Deliverable 3.5, CLASSIC Project.

Ghislain Putois, Romain Laroche, and Philippe Bretier. 2010. Enhanced monitoring tools and online dialogue optimisation merged into a new spoken dialogue system design experience. In *Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 185–192.

Antoine Raux, Brian Langner, Allan Black, and Maxine Eskenazi. 2003. LET'S GO: Improving Spoken Dialog Systems for the Elderly and Non-natives. In *Proceedings of Eurospeech*.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT*, pages 12–18.

David Suendermann, John Liscombe, and Roberto Pieraccini. 2010. Contender. In *Proceedings of IEEE SLT*, pages 330–335.

Joel R. Tetreault, Dan Bohus, and Diane J. Litman. 2007. Estimating the reliability of mdp policies: A confidence interval approach. In *Proceedings of HLT-NAACL*, pages 276–283.

Fan Yang and Peter A. Heeman. 2007. Exploring initiative strategies using computer simulation. In *Proceedings of Interspeech*, pages 106–109.