Hybridisation of Expertise and Reinforcement Learning in Dialogue Systems

Romain Laroche 12, Ghislain Putois 1, Philippe Bretier 1, Bernadette Bouchon-Meunier 23

1 Orange Labs, Issy les Moulineaux, France
2 Laboratoire d’Informatique de Paris VI, Paris, France
3 Centre National de Recherche Scientifique, Paris, France
romain.laroche@polytechnique.org

Abstract

This paper addresses the problem of introducing learning capabilities in industrial handcrafted automata-based Spoken Dialogue Systems, in order to help the developer to cope with his dialogue strategies design tasks. While classical reinforcement learning algorithms position their learning at the dialogue move level, the fundamental idea behind our approach is to learn at a finer internal decision level (which question, which words, which prosody, . . .). These internal decisions are made on the basis of different (distinct or overlapping) knowledge. This paper proposes a novel reinforcement learning algorithm that can be used to make a data-driven optimisation of such handcrafted systems. An experiment shows that the convergence can be up to 20 times faster than with Q-Learning.

Index Terms: SDS, reinforcement learning, adaptive dialogue

1. Introduction

Spoken Dialogue Systems (SDS) with learning capabilities have been deeply investigated these last ten years [1, 2, 3, 4]. The primary goal pursued by these studies was to reduce the development cost by automating the SDS design. They use Markov Decision Process (MDP) to learn the best dialogue move according to the current dialogue state. This approach has led to promising results but, as [5, 6] acknowledge, some obstacles are difficult to overcome. The high dimensionality of the dialogue states and actions is the main reason for all these troubles.

Firstly, the MDP sets (the state set and the action set) require a tight tuning, which can only be performed by a reinforcement learning expert and which is quite contrary to the original automation objective. Secondly, the learning process requires to gather huge dialogue corpora which can be problematic depending on the application. Finally, even when all those conditions are fulfilled, the automatically generated SDS reported in the literature are usually simple and hardly reach the dialogue quality of the fine tuned handcrafted ones.

Releasing noticeably the automation goal, while [7, 8] focused more specifically on the SDS robustness with POMDP, [9, 10] made several attempts to improve the learning convergence at the expense of even more MDP design. These papers propose to use various Hierarchical Reinforcement Learning algorithms (MAXQ [11] and HAM [12]) in order to help the learning with a defined hierarchy of strategies: macro-strategies (i.e. a strategy that is followed during several turns) and one-dialogue-turn strategy. The one-dialogue-turn strategies sequences are optimised in order to fulfil a macro-strategy and the transitions between macro-strategies are learnt too. While [9, 10] correctly stresses that in a dialogue, the system needs to handle macro-strategies, we rather insist on the fact that even a single dialogue move is already most of the time the composition of several strategic decisions (see subsection 2.1).

Another group of studies [13, 14] went even a bit further than [9, 10], and proposed to mix a handcrafted SDS with an MDP-based SDS. The goal here is completely different: to optimise the dialogue capabilities of a conventional SDS. In substance, the idea consists in handcrafting almost totally the SDS so that it provides a small collection of options among which the MDP-based SDS picks the action to generate to the user. Our paper completely agrees with this objective. We endeavour to improve these previous works by proposing a novel framework that works at the internal decision level (which question, which words, which prosody, . . .), instead of the dialogue move level. This enables us to drastically reduce the dimensionality of the learning and to consequently speed up the convergence.

Section 2 gives a detailed description of the problem and enumerates the constraints implied by the environment under consideration, which leads us to propose an extension of MDP called Module-Variable Decision Process (MVDP). Section 3 provides an algorithm for reinforcement learning in the MVDP framework: the Example-Based Reinforcement Learning (EBRL). Using a generalised problem, section 4 shows that the EBRL alleviates the “curse of dimensionality” and outperforms all the existing algorithms that could have been applied within the constraints defined in section 2. Finally, section 5 concludes the paper with the foreseen improvements.

2. Problem Description: Towards a New Model

2.1. Problem Constraints

Figure 1 shows a part of the design of an automated hotline dedicated to ADSL box installation. It reveals that the welcome message can be split up into three internal decisions: nature of the greeting/presentation, the help messages insertion or not and finally some choices of questions. These internal decisions can be made with a very localised view of the system state. Assuming all the enumerated information is available, the choice of the greeting message is dependent on the area of the call and the age of the caller and the choice of the help message is dependent basically on the SDS expertise of the caller. All these dependency definitions are part of the designer’s work and are not discussed in this article. This paper aims at providing a model and an algorithm for learning in such an environment. If we used a classical MDP at the dialogue move level, for the figure 1 example, the state set would be the crossproduct of all the dependency variables and the action set would also be the crossproduct of all the alternatives. The dimensionality would soar, compromising the convergence without an inordinate amount of dia-
2.2. Module-Variable Decision Process

A module is a processing unit that can choose an internal action according to its local variables. This leads us to the definition of the Module-Variable Decision Process (MVDP) framework \((M, V_m, A_m)\) where:

- \(M\) is the set of modules.
- \(\forall m \in M, V_m\) is the variable space used in module \(m\).
- \(\forall m \in M, A_m\) is the set of possible actions for \(m\).

In figure 1, the modules are the diamond-shaped boxes labelled 1, 2 and 3. For module 1, the possible internal actions are the 1.1, 1.2 and 1.3 transitions. The local states are not illustrated. In subsection 2.1, we told that the dependency of module 1 was the area of the call and the age of the caller. In such a design, \((\text{Paris, 30}) \in V_1\).

In a dialogue system, rewards are not synchronised with dialogue turn, some dialogue moves are composed of several internal decisions and some other dialogue moves are completely handcrafted (i.e. without using the learning capabilities of the system). As a consequence, internal decisions and rewards must be associated with an absolute timestamp in \(\mathbb{R}\) (typically expressed in dialogue turns or in seconds).

Each module has a policy. The policy governs the choices that are made when the system needs to make a decision in the module. A policy is a function from the variable space into the action space \(\pi_m : V_m \rightarrow A_m\). In order to build its policy, the module may generate a state-action value function \(Q_m : V_m \times A_m \rightarrow \mathbb{R}\) which intends to predict the dialogue-term reward given the local state and the chosen action. The exploitation policy is the one that aims to maximise the dialogue-term reward expectations after a given decision \(d\):

\[
r_d = \sum_k \gamma^{t_k - t_d} R_k
\]

Where \(\gamma \in [0, 1]\) is the discount factor, used in order to encourage the shortest path to a dialogue success, \(t_d\) is the time when decision \(d\) has been made and \(t_k > t_d\) is the time when reward \(R_k\) has been received.

3. Example-Based Reinforcement Learning

A (internal) decision \(d\) has the following features: the module \(m\) where \(d\) is made, the (local) state \(v\) reduced to the relevant information concerning \(d\), chosen action \(a\) and timestamp \(t\).

\[
d = (m, v, a, t) \in M \times V_m \times A_m \times \mathbb{R}
\]

An episode is the chain of decisions and rewards generated during a dialogue. The EBRL added value consists in avoiding to learn that an upfront decision is bad because the episode that tried it made further bad decisions. The EBRL rates an episode according to the system’s current policy. Thus, the EBRL may consider that an experience for a decision is not meaningful because it regards its further decisions as irrelevant. Concretely, the algorithm checks if the current policy would trigger the same choices if it had to repeat a recorded episode. This rating \(c_e(e) \in \mathbb{R}^+\) is called the compliance of an episode \(e\) with the policy \(\pi\). It represents the deviation of the episode \(e\) decisions from the policy \(\pi\).

The local \(\pi\)-compliance \(c_e(d_k)\) is computed thanks to the state-action value functions. The local compliance is the expected loss of performance implied by the chosen action \(a_k\) of decision \(d_k\), compared to the \(\pi\)-optimal one:

\[
c_e(d_k) = Q_m(v_k, a_k) - \sup_{a \in A_m} Q_m(v_k, a)
\]

Then, the global \(\pi\)-compliance \(c_e(e)\) of an episode \(e = (d_1, ..., d_{|e|})\) after a decision \(d_0\) is computed with a formula similar to that of the reward (equation (1)) is proposed:

\[
c_e(e) = \sum_{k=1}^{|e|} \beta^{t_k - t_0} c_e(d_k)
\]

Once corpus \(C = \{m_k, v_k, a_k, r_k, c_k\}_{k \in [1, n]}\) is generated, the Monte Carlo method [15] is adapted to accept weighted average on the returns. Therefore, \(Q\) expectation is computed as follows:

\[
C_{m,v,a} = \{r_k, c_k\} \text{ such that } \{m, v, a, r_k, c_k\} \in C
\]

\[
Q_m(v, a) = \sum_{r_k \in C_{m,v,a}} t_k e^{r_k}
\]

Where \(\tau\) is a parameter expressing the impact of the compliance on the weight for the averaging. The exponential reflects the intuition that several incompliances should multiply.

An EBRL implementation is provided by algorithm 1.
The episode starts in a random non-terminal state among the 124.

In order to reflect the dialogue domain interest of the approach, the model and the algorithm, (2) user simulations are too simple to grant a tangible evaluation of a dialogue system and (3) dialogue applications can be very unalike.

4.1. The Generalisation Problem

In the SDS literature, most papers test their learning algorithms with user simulations. However, we prefer to make our study on a simple generalisation problem for several reasons: (1) the generalisation problem provides a universal evidence of the interest of the approach, the model and the algorithm. (2) user simulations are too simple to grant a tangible evaluation of a dialogue system and (3) dialogue applications can be very unalike.

Therefore, we consider in this section a problem that generalises the example in figure 1. The system is given 27 alternative dialogue moves which correspond to the cross combinations of the three internal decision blocks proposing each 3 alternative local actions. Every dialogue turn, the same chain of decisions are made and the dialogue ends when:

- one of the three internal decision blocks is reached in a dialogue failure state. In that case, a negative reward $-1$ is received,
- or when all three internal decision blocks have reached their dialogue success states. In that case, a positive reward $+1$ is received.

Excluding the dialogue failure state (the dialogue ends as soon as they are accessed) but including the dialogue success state. In that case, a negative reward $-1$ is received, or when all three internal decision blocks have reached their dialogue success states. In that case, a positive reward $+1$ is received.

Excluding the dialogue failure state (the dialogue ends as soon as they are accessed) but including the dialogue success state. In that case, a negative reward $-1$ is received, or when all three internal decision blocks have reached their dialogue success states. In that case, a positive reward $+1$ is received.

4.2. Compared Algorithms

Three reinforcement learning methods are compared. For all of them, an $\epsilon$-greedy exploration is used with $\epsilon = 0.5 \times 0.998^{[E]}$.

The main goal of this problem is to study the gain obtained with the parallelisation of the problem thanks to the EBRL algorithm as illustrated in figure 2.

4.3. Comparison with the Naive Q-Learning Algorithm

As shown in figure 2, the complexity of the system is reduced by $\frac{124 \times 27 \times 5}{74} \approx 74$. The EBRL algorithm takes great advantage of the huge space reduction brought with the MVDP framework and figure 3 reveals that after making 500 episodes, which means basically 500 dialogue failures, NQL algorithm only succeeded in surviving one turn and a half longer on average, while EBRL has usually reached the optimal policy. The NQL algorithm actually requires 10000 episodes to reach the EBRL performance after 500 episodes.

4.4. Impact of $\tau$

Figure 4(a) shows the Monte Carlo control loop. Our algorithm improves the evaluation process by evaluating as well the past episodes quality for learning. As shown in figure 4(b), EBRL with the optimal $\tau$ value: $1.4$ performs significantly better than CMC ($\tau = 0$). Figure 4(c) shows that for moderate $\tau$ values, EBRL reliably outperforms the CMC algorithm. The fact that the compliance calculation is based on a sub-optimal policy in the early episodes explains the loss of performance when $\tau$ grows above $1.5$. In future works, we plan to study $\tau$ values that grows over time, along with the policy optimality.
5. Conclusion

This paper investigates reinforcement learning for SDS. It focuses more precisely on an expertise/learning hybridisation of the system. This approach leads to propose a novel decision process model: the Module-Variable Decision Process. Under this model, constraints prevents from using classical bootstrapping methods such as TD Learning. Then, a new algorithm is proposed: the Example-Based Reinforcement Learning. Eventually, an evaluation on a generalised problem shows that our algorithm outperforms all the well-known reinforcement learning algorithms.

The scope of this contribution is enhanced in three directions. First, figure 1 has for purpose to illustrate in an automaton design how internal decisions can combine and become a single dialogue move. But the MVDP framework does not rely on the automaton structure: the system gathers internal decisions independently of the internal structure, collects rewards, builds a corpus and uses it to learn a best practice policy. Second, EBRL can easily be adapted to accept any supervised learning algorithm instead of the weighted average computed with formula 6. Third, the application domain of EBRL can be extended to any handcrafted system requiring learning capabilities such as robotics or artificial intelligence design for video games.

As for perspectives, the algorithm has been integrated in a dialogue application for internet box installation help for applicative tests. Concerning the theoretical aspects, we plan to study the correspondence between the compliance and the average weights and to introduce uncertainty handling to the EBRL algorithm.

6. Acknowledgements

This research has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement number 216594 (CLASSIC project: www.classic-project.org).

7. References