Optimising a handcrafted dialogue system design

Conference Paper · January 2010

Source: DBLP

3 authors, including:

Romain Laroche
Microsoft Maluuba
58 PUBLICATIONS **185** CITATIONS
SEE PROFILE

Ghislain Putois
Voxygen SAS
22 PUBLICATIONS **80** CITATIONS
SEE PROFILE

Optimising a Handcrafted Dialogue System Design

Romain Laroche[1]*, Ghislain Putois*[1] *and Philippe Bretier*[1]

[1]Orange Labs, Issy-les-Moulineaux, France

romain.laroche@orange-ftgroup.com

Abstract

In the Spoken Dialogue System literature, all studies consider the dialogue move as the unquestionable unit for reinforcement learning. Rather than learning at the dialogue move level, we apply the learning at the design level for three reasons : 1/ to alleviate the high-skill prerequisite for developers, 2/ to reduce the learning complexity by taking into account just the relevant subset of the context and 3/ to have interpretable learning results that carry a reusable usage feedback. Unfortunately, tackling the problem at the design level breaks the Markovian assumptions that are required in most Reinforcement Learning techniques. Consequently, we decided to use a recent non-Markovian algorithm called Compliance Based Reinforcement Learning. This paper presents the first experimentation on online optimisation in dialogue systems. It reveals a fast and significant improvement of the system performance with by average one system misunderstanding less per dialogue.

Index Terms : Spoken Dialogue Systems, Reinforcement Learning, Online Learning, Hybrid System

1. Introduction

Spoken Dialogue Systems (SDS) with learning capabilities have been deeply investigated these last ten years [1]. The primary goal pursued by these studies is 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 some researchers [2] acknowledge, some obstacles are difficult to overcome (Reinforcement Learning skill prerequisite, simulated users, ...). The high dimensionality of the dialogue states and actions is the main reason for all these troubles. As a consequence, reducing this space complexity represents a big part of the scientific effort with for instance hierarchy of states [3] and summary of states [4]. In fact, these works endeavour to improve the system performance by inserting expert knowledge.

Other studies [5, 6] propose to mix a handcrafted SDS with an MDP based SDS. The goal 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. We completely agree with this objective. We endeavour to improve these previous works by using a novel framework that learns at the handcrafting design level (which question, which words, which prosody, ...), instead of the dialogue move level. This enables to drastically reduce the dimensionality of the learning and to consequently speed up the convergence [7].

Section 2 gives a detailed description of the problem and recalls the Compliance Based Reinforcement Learning algorithm [7] that we use for design optimisation. Then, section 3 describes the experimented system and the goal of the experimentation. Next, section 4 presents the experimentation results. Finally, section 5 concludes the paper with the next steps.

2. Problem Constraints and Theoretical Resolution

This section first recalls how conventional SDS are designed by developers. Then, it explains why the Markovian assumption cannot be maintained in this design environment. Thus, the Module-Variable-Decision Process is introduced. Finally, the section ends with the presentation of the Compliance-Based Reinforcement Learning : a reinforcement learning algorithm that does not use the Markovian assumption.

2.1. The conventional SDS design environment

Industry follows the VUI-completeness principle [8] : *"the behaviour of an application needs to be completely specified with respect to every possible situation that may arise during the interaction. No unpredictable user input should ever lead to unforeseeable behaviour"*. The SDS developers consider reliable the technologies, tools, and methodologies that help them to reach the VUI-completeness and to control it.

The graphical abstraction used for SDS design conforms to the general graph representation of finite state automata, with the difference that global and local variables (or context) enable to factorise several system states in a single node. A system state is therefore described by the automata node and the global context aggregating the global context with all the local contexts. Transitions relate to user inputs or to internal application events such as conditions based on internal information from the current dialogue state, from the back-end, or from the dialogue history. In that sense, dialogue design in the industry generally covers more than strict dialogue management, since its specification may indicate the type of spoken utterance expected from the user at each stage of the dialogue, up to the precise speech recognition model and parameter values to use, and the generation of the system utterance, from natural language generation to speech synthesis or audio recordings.

2.2. Problem Constraints

Most dialogue moves can be split up into several internal decisions that are represented by as many automata nodes : type of feedback, insertion of contextual help or presentation of additional information and choices of questions. These internal decisions are conventionally specified with handcrafted rules designed from a very localised view of the system state. For instance, the system's decision to welcome the user with "good morning" or "good afternoon" will depend solely on the current time at the place from where the user calls. Another more complex example : the decision to insert one help message depends

on the expected expertise of the user. It is easy to understand that this expected expertise depends on the service users' average expertise and on the number of errors that occurred during the current dialogue. However, it is very complex for the developer to estimate the users average expertise or to design a function between those context elements and the decision to make. This is an example where the decision has to be based on statistics. The developer designs the decision alternatives and the relevant information to take into account for making this decision. Learning at a decision level complies with the way developers are currently designing their application.

Learning at a decisional level is also motivated by the learning improvement demonstrated on simulated examples in [7]. Indeed, it enables to use for each decision only the precise relevant context, instead of taking into account the whole dialogue context. This leads to a space reduction and therefore to a faster convergence.

In addition to providing a tool for experimenting several alternatives and optimising the system's behaviour, this approach offers reporting capabilities. In fact, it delivers a usage feedback inside the very dialogue automata the developer designed in the first place. Contrary to MDP-based methods that flourish in the literature, our method enables the explanation and thus the understanding of what the system learnt. Then, this inferred knowledge can be reused in a another context or even another dialogue application. This ability to generate expert knowledge acquired with online learning is new in SDS literature.

Unfortunately, in spite of these advantages, our approach has a strong technical shortcoming : it breaks the Markovian assumptions. Indeed, at a time t, the decision process state s_t is described by the automata node m_t and the local context v_t taken into account for making the decision. The next decision process state s_{t+1} cannot be forecast because its local context v_{t+1} is generally not included into s_t. As the decision process is non-Markovian, in a given decision process state, we cannot assume to be able to anticipate the next reached decision process state. As a consequence, we cannot use any bootstrapping method [9], *i.e.* any method that updates expectation estimates on the basis of the estimates of the following decision process states. The fastest reinforcement learning algorithms use bootstrapping, such as Dynamic Programming or Temporal Difference Learning.

2.3. Module-Variable Decision Process

This subsection recalls a recent framework [7] for learning at a decision level. Contrarily to MDP that consider the system state as a whole, in this framework, the system global state is not represented in the decision process. The decision process states are the locally relevant subset of the information included in the system state. Further in this paper, a *state* refers to the decision process state and the global system state concept is abandoned.

A *module* is the terminology we use for an automata node. For practical purpose, it is a processing unit that can execute an internal *action* according to its local *context*. This leads 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 local context used in module m.
- $\forall m \in M$, A_m is the set of possible actions for m.

In our previous example, we called s_t the decision process state at time t. This state is a tuple made of the module m_t accessed at time t and its local context v_t. Each module has a *policy*. The policy governs the choices that are made when reaching the corresponding automata node given a local context. A policy is a function from the context space into the action space $\pi_m : V_m \mapsto A_m$. In order to build its policy, the module may generate a *state-action value function* $Q_m : V_m \times A_m \mapsto \mathbb{R}$ which intends to predict the dialogue-term reward given the local context and the chosen action. The exploitation policy aims to maximise the dialogue-term reward expectations after a given decision d :

$$r_d = \sum_k \gamma^{t_k - t_d} R_k \tag{1}$$

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.

2.4. Compliance Based Reinforcement Learning

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

$$d = (m, v, a, t) \in M \times V_m \times A_m \times \mathbb{R} \tag{2}$$

From the reinforcement learning algorithm point of view, an *episode* is the chain of decisions and rewards generated during a dialogue. The CBRL idea consists in avoiding to learn that an upfront decision is bad because the episode that tried it made further bad decisions. The basic idea of the algorithm is to avoid that an episode (d_a, d_b) where d_b is bad (and thus engendered a low reward) leads the system to learn that d_a is bad too based on this episode poor performance. In order to prevent this, the algorithm rates to what extent each episode is reliable for learning. Thus, the CBRL may consider that an episode should not be taken into account for evaluating a decision d_a because the further decisions are considered not good enough. This rating $c_\pi(e) \in \mathbb{R}^-$ is called the *compliance* of an episode e with the policy π. It represents the deviation of the episode e decisions from the policy π and it is computed as follows :

$$c_\pi(e) = \sum_{k=1}^{|e|} \gamma^{t_k - t_0} \left(Q_{m_k}(v_k, a_k) - \sup_{a \in A_{m_k}} Q_{m_k}(v_k, a) \right) \tag{3}$$

This compliance measures how well a decision has been evaluated by computing how compliant the further decisions of the episode are according to the current policy. Once the decision corpus $C = \{m_k, v_k, a_k, r_k, c_k = c_\pi(e_k)\}$ is generated (e_k is the episode after making decision d_k), the Monte Carlo method [9] is adapted to accept weighted average on the returns. Therefore, Q expectation is computed as follows :

$$C_{m,v,a} = \{r_k, c_k\} \text{ with } \{m, v, a, r_k, c_k\} \in C \tag{4}$$

$$Q_m(v, a) = \frac{\sum_{\{r_k, c_k\} \in C_{m,v,a}} r_k e^{\tau c_k}}{\sum_{\{r_k, c_k\} \in C_{m,v,a}} e^{\tau c_k}} \tag{5}$$

Where τ is a parameter expressing the impact level of the compliance on the weights.

3. Experimentation Settings

Our experimentation is a proof of concept on a small set of users before implementing these learning capabilities on a commercial system receiving dozens of thousands calls per month. In order to observe an improvement of the system after less than 200 calls, we had to reduce the alternative sets to four design points and also to limit the local context of each point to the empty set. With these limitations, an MDP system would probably have performed as well. The goal of this experimentation was not to show the performance superiority of the CBRL approach, which has been proven in a previous paper [7]. The goal was to prove that it was possible to improve dramatically a system's performance by optimising it through a simple and reduced set of alternatives.

We insist that the goal was not to learn a reusable policy as with batch learning which are trained with simulated users (even if it's technically possible). Indeed, our algorithm is designed for online learning, *i.e.* learning during runtime and our experimentation shows how a commercial system would improve during its lifetime, on the basis of its interactions with real customers.

The experimented system helps the user to install her DSL-box (called Livebox). The system and the evaluation forms were in French only, but for an easier understanding, all the information is here translated into English.

3.1. Implementation

As carrying out the Livebox installation by phone is a bit clunky, the system first tries to find another means to help the user with her Livebox installation. As a result, during the first part of the service, the system asks whether the user wants a technician to install her Livebox (if she does, the user is directed to the appointment scheduling service), whether the user is at home (if not, the user is asked to call again when she is) and whether the user has access to the internet (if she does, the system provides her with a URL where she can find the Livebox installation process). Once the system has checked that it could not bypass the installation process, the hardware installation process is described to the user. The installation process involves the plugging of the power cable, the DSL cable, the Ethernet cable and the DSL filters.

3.2. Proposed Alternatives

Instead of designing a completely deterministic service, we experimented several alternatives at four locations (or modules in the MVDP framework, see section 2.3) in the design. As we knew that the dialogue corpus would be limited to 160 units, we decided not to condition on context space V_m. The set of alternatives A_m are the following ones :

Orange Labs state-of-the-art unit selection speech synthesizer[1] was used with different acoustic inventories of the same professional female speaker to generate acoustic variants of the greeting message : neutral, calm and expressive.

Two orders for the bypassing questions were experimented : 1) home/internet/technician and 2) technician/home/internet.

Two natural language generation variants of the "are you at home" message were tested : a directive variant ("I would like to know if you are at home now.") and an interrogative variant ("Are you now at home ?"). Three speaking style TTS variants were tested for the interrogative variant : neutral, expressive and calm.

[1] demonstrator available at http ://tts.elibel.tm.fr

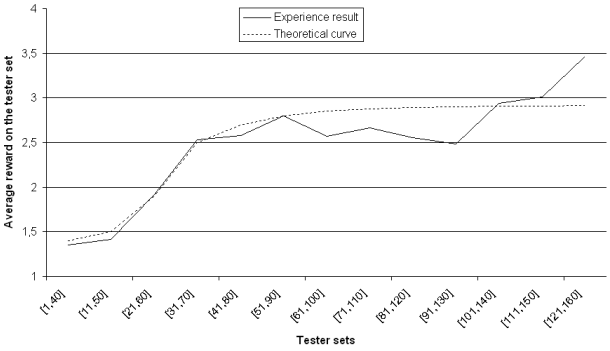

FIG. 1 – Overall performance evolution

During the installation process, there are a lot of information to present to the user : how they should plug the power cable, the DSL cable, the Ethernet cable, the ToIP adapter (which is not used in the installation process) and the DSL filters. There is no straightforward relevant order for these procedures. For instance, we have considered that the user might be more comfortable if the system first ask her to identify each element, before undertaking the plugging. Eventually, five different strategies have been tested.

4. Evaluation and Analysis

The system evaluation was completed using 40 subjects each performing four scenarios, corresponding to each three bypassing questions plus the full installation procedure.

4.1. System Auto-evaluation

4.1.1. Settings

The System auto-evaluation is the metrics used as rewards for the CBRL. In this experiment, they were task-based as follows :

- -1 for each ASR/SLU reject or time-out
- -10 for a hang-up or after an unsuccessful Livebox installation
- +5 after providing the user with an alternative way to install her Livebox.
- +10 after a successful Livebox installation

4.1.2. Overall Performance Evolution

The experimental results in figure 1 show that the learning algorithm choices triggered an overall dialogue performance improvement through time in a window of 40 dialogues.

The flat performance at the first stage can be explained by the exploration policy that let the first testers experience an almost fully exploration strategy, to be able to rate the fully exploratory policy. Then, around abscissa points [11,50] and [31,70], there follows a strong performance improvement probably due to what the learning algorithm learnt. The small decreasing of the plot around the [81,120] abscissa is probably due to testers that were performing under average. Similarly, the very high average rewards obtained at the end of the experimentation should be put into perspective with the fact that no learning has been done at this time, and this improvement can only be explained by users performing over the average. The dotted curve shows how we can extrapolate the system's performance

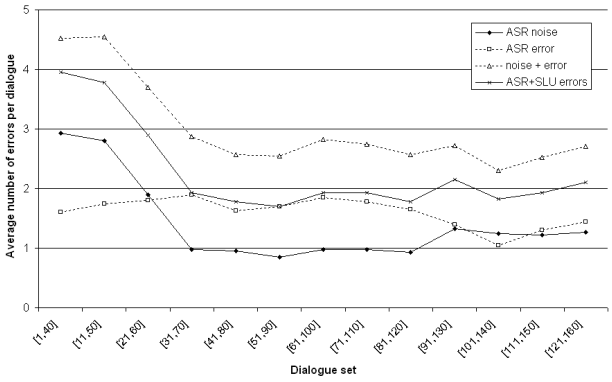

FIG. 2 – Evolution of the error rate based KPIs

expectations over time. This results shows that very similar systems can vary a lot in performance. It also shows that our algorithm significantly improved a handcrafted system in a very short time.

4.2. Objective Evaluation

In order to prove that reward's optimisation led to a real objective improvement of the system, we made two sets of objective Key Performance Indicators (KPI) : the duration based KPI such as call duration in seconds and number of dialogue turns, and the error rate based KPI such as ASR and SLU error rate. The duration based KPI were obtained automatically, while the error rate KPI were based on manual dialogue annotations.

Experimentation showed that the duration based KPI are not significantly connected with the dialogue performance as defined with the system rewards.

Concerning the error-based KPI, we considered ASR noise (ASR errors caused by the surrounding noise, the Livebox manipulation triggered a lot of them), ASR errors, ASR noise+error (the sum of the two previous KPI) and ASR+SLU errors (errors after the ASR+SLU chain).

Figure 2 shows that the main KPI to be affected by the system's learning is the ASR noise. That is how the choice of the best alternative expresses its improvement. To the contrary, the ASR errors do not look much affected by the learning. The error rates for ASR and ASR+SLU are directly influenced by the ASR noise curve and eventually the average number of errors per dialogue is divided by 2, which constitutes a strong result.

5. Conclusion

This paper presented the first experimentation in a real Spoken Dialogue System optimising online. It recalled the MVDP framework and the CBRL algorithm that were used for the application design. An evaluation on the Livebox installation applicatin was made : auto-evaluation of the system and objective evaluation. The auto-evaluation results showed that the break of the Markovian assumption did not endanger the convergence of the learning algorithm. The objective evaluation confirmed that the reward optimisation led to a dialogue error cut, and that the system really performed better at the end of the experimentation than at the beginning.

This paper proved the validity of our approach for optimisation in SDS with a significant error reduction along the learning. In addition, the experimentation showed other less quantifiable advantages of our approach, when compared to dialogue move optimisation. First, it is easier to implement. The applicative implementation does not require any technical skills. The developer just has to define the alternatives, the local context and the rewards, as opposed to dialogue turn based learning that suppose a summary/hierarchy of the dialogue state set. Second, although dialogue move learning only provides a general optimisation that cannot be interpreted or explained, our approach provides a valuable usage return for the application designers. In addition to that, a call flow monitoring tool has been integrated to the design studio, so that the developers or project managers can overview how each alternative performed and how they correlate to key performance indicators.

6. Acknowledgment

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

[1] O. Lemon and O. Pietquin, "Machine learning for spoken dialogue systems," in *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, August 2007, pp. 2685–2688.

[2] T. Paek and R. Pieraccini, "Automating spoken dialogue management design using machine learning : An industry perspective," *Speech Communication*, vol. 50, pp. 716–729, 2008.

[3] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Reinforcement learning of dialogue strategies with hierarchical abstract machines," in *Proceedings of IEEE/ACL Workshop on Spoken Language Technology (SLT)*, December 2006.

[4] J. D. Williams and S. Young, "Scaling up POMDPs for dialog management : The summary POMDP method," in *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, 2005, pp. 177–182.

[5] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing Dialogue Management with Reinforcement Learning : Experiments with the NJFun System." *Journal of Artificial Intelligence Research*, vol. 16, pp. 105–133, 2002. [Online]. Available : http ://www.jair.org/media/859/live-859-1983-jair.pdf

[6] J. D. Williams, "The best of both worlds : Unifying conventional dialog systems and POMDPs," in *International Conference on Speech and Language Processing*, 2008.

[7] R. Laroche, G. Putois, P. Bretier, and B. Bouchon-Meunier, "Hybridisation of expertise and reinforcement learning in dialogue systems," in *Proceedings of Interspeech. Special Session : Machine Learning for Adaptivity in Spoken Dialogue*, Brighton (United Knigdom), September 2009.

[8] R. Pieraccini and J. Huerta, "Where do we go from here ? research and commercial spoken dialog systems," in *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, L. Dybkjaer and W. Minker, Eds., 2005, pp. 1–10.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998. [Online]. Available : http ://www.amazon.ca/exec/obidos/redirect ?tag=citeulike09-20&amp ;path=ASIN/0262193981