Human-Machine Dialogue as a Stochastic Game

Merwan Barlier[1,2]
[1]NaDia Team
Orange Labs
merwan.barlier@orange.com

Julien Perolat[2]
[2]Univ. Lille - CRIStAL lab
SequeL team
julien.perolat@univ-lille1.fr

Romain Laroche
NaDia Team
Orange Labs
romain.laroche@orange.com

Olivier Pietquin[2,3]
[3]Institut Universitaire de France
IUF
olivier.pietquin@univ-lille1.fr

Abstract

In this paper, an original framework to model human-machine spoken dialogues is proposed to deal with co-adaptation between users and Spoken Dialogue Systems in non-cooperative tasks. The conversation is modeled as a Stochastic Game: both the user and the system have their own preferences but have to come up with an agreement to solve a non-cooperative task. They are jointly trained so the Dialogue Manager learns the optimal strategy against the best possible user. Results obtained by simulation show that non-trivial strategies are learned and that this framework is suitable for dialogue modeling.

1 Introduction

In a Spoken Dialogue System (SDS), the Dialogue Manager (DM) is designed in order to implement a decision-making process (called *strategy* or *policy*) aiming at choosing the system interaction moves. The decision is taken according to the current interaction context which can rely on bad transcriptions and misunderstandings due to Automatic Speech Recognition (ASR) and Spoken Language Understanding (SLU) errors. Machine learning methods, such as Reinforcement Learning (RL) (Sutton and Barto, 1998), are now very popular to learn optimal dialogue policies under noisy conditions and inter-user variability (Levin and Pieraccini, 1997; Lemon and Pietquin, 2007; Laroche et al., 2010; Young et al., 2013). In this framework, the dialogue task is modeled as a (Partially Observable) Markov Decision Process ((PO)MDP), and the DM is an RL-agent learning an optimal policy. Yet, despite some rare examples, RL-based DMs only consider task-oriented dialogues and stationary (non-adapting) users.

Unfortunately, (PO)MDP are restricted to model game-against-nature problems (Milnor, 1951). These are problems in which the learning agent evolves in an environment that doesn't change with time and acts in a totally disinterested manner. (PO)MDP-based dialogue modeling thus applies only if 1) the user doesn't modify his/her behavior along time (the strategy is learned for a stationary environment) and 2) the dialogue is task-oriented and requires the user and the machine to positively collaborate to achieve the user's goal.

The first assumption doesn't hold if the user adapts his/her behavior to the continuously improving performance of a learning DM. Some recent studies have tried to model this co-adaptation effect between a learning machine and a human (Chandramohan et al., 2012b) but this approach still considers the user and the machine as independent learning agents. Although there has already been some few attempts to model the "co-evolution" of human machine interfaces (Bourguin et al., 2001), this work doesn't extend to RL-based interfaces (automatically learning) and is not related to SDS.

More challenging situations do also arise when the common-goal assumption doesn't hold either, which is the case in many interesting applications such as negotiation (El Asri et al., 2014), serious games, e-learning, robotic co-workers *etc*. Especially, adapting the MDP paradigm to the case of negotiation dialogues has been the topic of recent works. In (Georgila et al., 2014), the authors model the problem of negotiation as a Multi-Agent Reinforcement Learning (MARL) problem. Yet, this approach relies on algorithms that are treat-

Proceedings of the SIGDIAL 2015 Conference, pages 2–11,
Prague, Czech Republic, 2-4 September 2015.

ing the multi-player issue as a non-stationnarity problem (*e.g.* WoLF-PHC (Bowling and Veloso, 2002)). Each agent is assumed to keep a stable interaction policy for a time sufficiently long so that the other agent can learn it's current policy. Otherwise, there is no convergence guarantees. Another major issue with these works is that noise in the ASR or NLU results is not taken into account although this is a major reason for using stochastic dialogue models. In (Efstathiou and Lemon, 2014), the authors follow the same direction by considering both agents as acting in a stationary MDP.

In this paper, we propose a paradigm shift from the now state-of-the-art (PO)MDP model to Stochastic Games (Patek and Bertsekas, 1999) to model dialogue. This model extends the MDP paradigm to multi-player interactions and allows learning jointly the strategies of both agents (the user and the DM), which leads to the best system strategy in the face of the optimal user/adversary (in terms of his/her goal). This paradigm models both co-adaptation and possible non-cooperativness. Unlike models based on standard game theory (Caelen and Xuereb, 2011), Stochastic Games allow to learn from data. Especially, departing from recent results (Perolat et al., 2015), we show that the optimal strategy can be learned from batch data as for MDPs (Pietquin et al., 2011). This means that optimal negotiation policies can be learnt from non-optimal logged interactions. This new paradigm is also very different from MARL methods proposed in previous work (Chandramohan et al., 2012b; Georgila et al., 2014; Efstathiou and Lemon, 2014) since optimization is jointly performed instead of alternatively optimizing each agent, considering the other can stay stationary for a while. Although experiments are only concerned with purely adversarial tasks (Zero-Sum games), we show that it could be naturally extended to collaborative tasks (general sum games) (Prasad et al., 2015). Experiments show that an efficient strategy can be learned even under noisy conditions which is suitable for modeling realistic human-machine spoken dialogues.

2 Markov Decision Processes and Reinforcement Learning

As said before, human-machine dialogue has been modeled as an (PO)MDP to make it suitable for automatic strategy learning (Levin and Pieraccini, 1997; Young et al., 2013). In this framework, the dialogue is seen as a turn-taking process in which two agents (a user and a DM) interact through a noisy channel (ASR, NLU) to exchange information. Each agent has to take a decision about what to say next according to the dialogue context (also called dialogue state). In this section, MDPs (Puterman, 1994) and RL (Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996) are briefly reviewed and formally defined which will help switching the Stochastic Games in Section 3.

2.1 Markov Decision Processes

Definition 2.1. A Markov Decision Process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ where: $\mathcal{S}$ is the discrete set of environment states, $\mathcal{A}$ the discrete set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ the state transition probability function and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function. Finally, $\gamma \in [0, 1)$ is a discount factor.

At each time step, the RL-agent acts according to a policy π, which is either deterministic or stochastic. In the first case, π is a mapping from state space to action space : $\pi : \mathcal{S} \rightarrow \mathcal{A}$, while in the latter, π is a probability distribution on the state-action space $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. Policies are generally designed to maximize the value of each state, *i.e.* the expected discounted cumulative reward: $\forall s \in \mathcal{S}, V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) | s_0 = s]$. Let $\mathcal{V}$ be the space of all possible value functions. The optimal value function V^* is the only value function such that: $\forall V \in \mathcal{V}, \forall s \in \mathcal{S}, V^* \geq V$. The following result, proved in (Puterman, 1994), is fundamental in the study of MDPs:

Theorem 2.1. *Let M be an MDP. Its optimal value function V^* exists, is unique and verifies:*

$$\forall s \in \mathcal{S}, V^*(s) = \max_{a \in \mathcal{A}} \Big(r(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V^*(s') \Big)$$

Furthermore, one can always find a deterministic policy π^* inducing V^*.

The function $Q_\pi : (s, a) \mapsto r(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V_\pi(s')$ is called Q-function. We thus have: $\pi^*(s) = \text{argmax}_a Q_{\pi^*}(s, a) = \text{argmax}_a Q^*(s, a)$.

2.2 Reinforcement Learning

In many cases, transition and reward functions are unknown. It is thus not possible to compute values

nor Q-Functions, the RL-agent learns an approximation by sampling through actual interactions with the environment. The set of techniques solving this problem is called *Reinforcement Learning*.

For instance the *Q-Learning algorithm* (Watkins and Dayan, 1992) approximates, at each time step, the optimal Q-Function and uses the following update rule:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha[r_{t+1}(s_t, a_t) + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$$

It can been shown that, under the assumption that $\sum \alpha = \infty$ and $\sum \alpha^2 < \infty$ and that all states are visited infinitely often, Q-values converge towards the optimal ones. Thus, by taking at each state the action maximizing those values, one finds the optimal policy. There are batch algorithms solving the same problem among which Fitted-Q (Gordon, 1999; Ernst et al., 2005).

3 Stochastic Games

Stochastic Games (Filar and Vrieze, 1996; Neyman and Sorin, 2003), introduced in (Shapley, 1953), are a natural extension of MDPs to the Multi-Agent setting.

3.1 Definitions

Definition 3.1. A discounted Stochastic Game (SG) is a tuple $\langle \mathcal{D}, \mathcal{S}, \mathbf{A}, \mathcal{T}, \mathbf{R}, \gamma \rangle$ where: $\mathcal{D} = \{1, ..., n\}$ represents the set of agents, $\mathcal{S}$ the discrete set of environment states, $\mathbf{A} = \times_{i \in \mathcal{D}} \mathcal{A}_i$ the joint action set, where for all $i = 1, ..., n$, $\mathcal{A}_i$ is the discrete set of actions available to the i^{th} agent, $\mathcal{T} : \mathcal{S} \times \mathbf{A} \times \mathcal{S} \rightarrow [0, 1]$ the state transition probability function, $\mathbf{R} = \times_{i \in \mathcal{D}} \mathcal{R}_i$ the joint reward function, where for all $i = 1, ..., n$, $\mathcal{R}_i : \mathcal{S} \times \mathbf{A} \rightarrow \mathbb{R}$ is the reward function of agent i. Finally, $\gamma \in [0, 1)$ is a discount factor.

An agent i chooses its actions according to some *strategy* σ_i, which is in the general case a probability distribution on i's state-action space. If the whole space of agents is considered, we speak about the *joint strategy* $\boldsymbol{\sigma}$. The notation σ_{-i} represents the joint strategy of all agents except i.

This definition is general, every 'MDP' in which multiple agents interact may be interpreted as a Stochastic Game. It is therefore useful to introduce a taxonomy. A game where there are only two players and where the rewards are opposite (*i.e.* $\mathcal{R}_1 = -\mathcal{R}_2$) is called *Zero-Sum Game*. Conversely, a *Purely Cooperative Game* is a game where all the agents have the same reward (*i.e.* $\forall i \in \mathcal{D}, \mathcal{R}_i = \mathcal{R}$). A game which is neither Zero-Sum nor Purely Cooperative is said to be *General-Sum*.

3.2 Best Response

In all environments, agents learn by acting according to what has previously been learned. In other words, agents adapt to an environment. This is also valid in a multi-agent scenario, if agent i wants to learn about agent j, it will act according to what has previously been learned about j. But conversely, if j wants to learn about agent i, it will act according to what it knows about i. We say that agents co-adapt. Co-adapation is, due to this feedback loop, an intrinsically non-stationary process. An algorithm converges if it converges to stationary strategies.

Each agent acts in order to maximize its expected discounted cumulative reward, also called the discounted value of the joint strategy $\boldsymbol{\sigma}$ in state s to player i : $V_i(s, \boldsymbol{\sigma}) = E[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{\sigma})]$. The Q-function is then defined as (Filar and Vrieze, 1996):

$$Q(s, \boldsymbol{\sigma}, \mathbf{a}) = R(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \mathbf{a}, s') V(s', \boldsymbol{\sigma})$$

This value function depends on the opponents' strategies. It is therefore not possible to define in the general case a strategy *optimal* against every other strategy. A Best Response is an optimal strategy given the opponents ones.

Definition 3.2. Agent i plays a Best Response σ_i against the other players' joint strategy σ_{-i} if σ_i is optimal given σ_{-i}. We write $\sigma_i \in BR(\sigma_{-i})$.

Best Response induces naturally the following definition:

Definition 3.3. The strategy profile $\{\sigma_i\}_{i \in \mathcal{D}}$ is a Nash Equilibrium (NE) if for all $i \in \mathcal{D}$, we have $\sigma_i \in BR(\sigma_{-i})$.

It is interesting to notice that in a single-player game, Nash Equilibrium strategies match the optimal policies defined in the previous section.

The existence of Nash Equilibria in all discounted Stochastic Games is assured by the following theorem (Filar and Vrieze, 1996):

Theorem 3.1. *In a discounted Stochastic Game G, there exists a Nash Equilibrium in stationary strategies.*

Two remarks need to be introduced here. First, nothing was said about uniqueness since in the general case, there are many Nash Equilibria. Equilibrium selection and tracking may be a big deal while working with SGs. Second, contrarily to the MDP case, there may be no deterministic Nash Equilibrium strategies (but only stochastic).

3.3 The Zero-Sum Case

There are two ways to consider a Zero-Sum Stochastic Game: one can see two agents aiming at maximizing two opposite Q-functions or one can also see only one Q-function, with the first agent (called the *maximizer*) aiming at maximizing it and the second one (the *minimizer*) aiming at minimizing it. One can prove (Patek and Bertsekas, 1999), that if both players follow those maximizing and minimizing strategies, the game will converge towards a Nash Equilibrium, which is the only one of the game. In this case, thanks to the Minmax theorem (Osborne and Rubinstein, 1994), the value of the game is (with player 1 maximizing and player 2 minimizing):

$$\begin{aligned} V^* &= \max_{\sigma_1} \min_{\sigma_2} V(\sigma_1, \sigma_2) \\ &= \min_{\sigma_2} \max_{\sigma_1} V(\sigma_1, \sigma_2) \end{aligned}$$

As we will see later, the existence of this unique value function for both player is helpful for finding efficient algorithms solving zero-sum SGs.

4 Algorithms

Even if the field of Reinforcement Learning in Stochastic Games is still young and guaranteed Nash Equilibrium convergence with tractable algorithms is, according to our knowledge, still an open problem, many algorithms have however already been proposed (Buşoniu et al., 2008), all with strengths and weaknesses.

Reinforcement Learning techniques to solve Stochastic Games were first introduced in (Littman, 1994). In his paper, Littman presents minimax-Q, a variant of the Q-Learning algorithm for the zero-sum setting, which is guaranteed to converge to the Nash Equilibrium in self-play. He then extended his work in (Littman, 2001) with Friend-or-Foe Q-Learning (FFQ), an algorithm assured to converge, and converging to Nash Equilibria in purely cooperative or purely competitive settings. The authors of (Hu and Wellman, 2003) were the first to propose an algorithm for general-sum Stochastic Games. Their algorithm, Nash-Q, is also a variant of Q-Learning able to allow the agents to reach a Nash Equilibrium under some restrictive conditions on the rewards' distribution. In the general case, they empirically proved that convergence was not guaranteed any more. (Zinkevich et al., 2006) proved by giving a counter-example that the Q-function does not contain enough information to converge towards a Nash Equilibrium in the general setting.

For any known Stochastic Game, the Stochastic Tracing Procedure algorithm (Herings and Peeters, 2000) finds a Nash Equilibrium of it. The algorithm proposed in (Akchurina, 2009) was the first learning algorithm converging to an approximate Nash Equilibrium in all settings (even with an unknown game). Equilibrium tracking is made here by solving at each iteration a system of ordinary differential equations. The algorithm has no guaranty to converge toward a Nash Equilibrium even however, it seems empirically to work. Finally, (Prasad et al., 2015) presented two algorithms converging towards a Nash Equilibrium in the General-Sum setting: one batch algorithm assuming the complete knowledge of the game and an on-line algorithm working with simulated transitions of the Stochastic Game.

In this paper we will use two algorithms which are reviewed hereafter: WoLF-PHC (Bowling and Veloso, 2002) and AGPI-Q (Perolat et al., 2015).

4.1 WoLF-PHC

WoLF-PHC is an extension of the Q-learning algorithm allowing probabilistic strategies. It considers independent agents evolving in an environment made non-stationary by the presence of the others. In such a setting, the aim of the agents is not to find a Nash Equilibrium (it is therefore not an SG algorithm) but to do as good as possible in this environment (and as a consequence, it may lead to a Nash Equilibrium). The algorithm is based on the following idea: convergence shall be facilitated if agents learn quickly to adapt when they are sub-optimal and learn slowly when they are near-optimal (in order to let the other agents adapt to this strategy).

Q-values are updated as in Q-learning and the probability of selecting the best action is incrementally increased according to some (variable) learning rate δ, which is decomposed into two learning rates δ_L and δ_W, with $\delta_L > \delta_W$. The

policy update is made according to δ_L while losing and to δ_W while winning.

To determine if an agent is losing or winning, the expected value of its actual strategy π, is compared to the expected value of the average policy $\overline{\pi}$. Formally, an agent is winning if $\sum_a \pi(s,a)Q(s,a) > \sum_a \overline{\pi}(s,a)Q(s,a)$ and losing otherwise.

In the general case, convergence is not proven and it is even shown on some toy-examples that sometimes, the algorithm does not converge (Bowling and Veloso, 2002).

4.2 AGPI-Q

Approximate Generalized Policy Iteration-Q, or AGPI-Q (Perolat et al., 2015), is an extension of the Fitted-Q (Gordon, 1999; Ernst et al., 2005) algorithm solving Zero-Sum Stochastic Games in a batch setting. At the initialization step, N samples (s, a_1, a_2, r, s') and a Q-function (for instance, the null function) are given. The algorithm consists then in K iterations, each of them composed of two parts : a *greedy part* and an *evaluation part*. The algorithm provides then at each iteration a better approximation of the Q-function.

Let $j = (s^j, a^j, b^j, r^j, s'^j)$ be N collected samples. At time step $k+1$, the *greedy part* consists of finding the maximizer's maxminimizing action $\overline{a}$ of the matrix game defined by $Q_k^j(s'^j, a^j, b^j)$. In our case, a turn-based setting, this involves finding a maximum. Then, during the *evaluation part*, since the second agent plays a minimizing strategy, the following value is computed: $Q^j = r + \gamma \min_b Q_k^j(s'^j, \overline{a}^j, b)$. At each iteration, the algorithm returns the Q-function Q_{k+1} fitting at best these values over some hypothesis space.

5 Dialogue as a Stochastic Game

Dialogue is a multi-agent interaction and therefore, it shall be considered as such during the optimization process. If each agent (*i.e.* the user and the DM) has its own goals and takes its decisions to achieve them, it sounds natural to model it as an MDP. In traditional dialogue system studies, this is only done for one conversant over two. Since (Levin and Pieraccini, 1997; Singh et al., 1999), only the DM is encoded as an RL agent, despite rare exceptions (Chandramohan et al., 2011; Chandramohan et al., 2012b; Chandramohan et al., 2012a)). The user is rather considered as a stationary agent modeled as a Bayesian network (Pietquin, 2006) or an agenda-based process (Schatzmann et al., 2007), leading to modeling errors (Schatztnann et al., 2005; Pietquin and Hastie, 2013).

At first sight, it seems reasonable to think that if two RL agents, previously trained to reach an optimal strategy, interact with each other, it would result in "optimal" dialogues. Yet, this assertion is wrong. Each agent would be optimal given the environment it's been trained on, but given another environment, nothing can be said about the learnt policy. Furthermore, if two DMs are trained together with traditional RL techniques, no convergence is guaranteed since, as seen above, non-stationarities emerge. Indeed, non-stationarity is not well managed by standard RL methods although some methods can deal with it (Geist et al., 2009; Daubigney et al., 2012) but adaptation might not be fast enough.

Jointly optimizing RL-agents in the framework of Stochastic Games finds a Nash Equilibrium. This guarantees both strategies to be optimal and this makes a fundamental difference with previous work (Chandramohan et al., 2012b; Georgila et al., 2014; Efstathiou and Lemon, 2014).

In the next section, we illustrate how dialogue may be modeled by a Stochastic Game, how transitions and reward functions depend on the policy of both agents. We propose now a Zero-Sum dialogue game where agents have to drive efficiently the dialogue to gather information quicker than their opponent. In this example, human user (Agent 1) and DM (Agent 2) are modeled with MDPs: each of them has a goal encoded into reward functions $\mathcal{R}_1$ and $\mathcal{R}_2$ (they may depend on the *joint action*).

5.1 A Zero-Sum Dialogue Game

The task involves two agents, each of them receives a random secret number and aims at guessing the other agent's number. They are adversaries: if one wins, the other one loses as much.

To find the secret number out, agents may perform one of the following actions: `ask`, `answer`, `guess`, `ok`, `confirm` and `listen`.

During a dialogue turn, the agent asking the question is called the guesser and the one answering is the opponent. To retrieve information about the opponent's hidden number, the guesser may `ask` if this number is smaller or greater than some other number. The opponent is forced to `answer`

the truth. To show that it has understood the answer, the agent says `ok` and releases then the turn to its adversary, which endorses the guesser's role.

Agents are not perfect, they can misunderstand what has been said. This simulates ASR and NLU errors arising in real SDSs. They have an indicator giving a hint about the probability of having well understood (a confidence level). They are however never certain and they may answer a wrong question, *e.g.* in the following exchange :

- Is your secret number greater than x ?

- My number is greater than y.

When such an error arises, Agent 1 is allowed to ask another question instead of just saying `ok`. This punishment is harsh for the agent which misunderstood, it is almost as if it has to pass its turn. Another dialogue act is introduced to deal with such situations. If an agent is not sure, it may ask to `confirm`. In this case, Agent 1 may ask its question again. To avoid abuses, *i.e.* infinitely ask for a confirmation, this action induces a cost (and therefore a gain for the opponent).

If an agent thinks that it has found the number out, it can make a `guess`. If it was right, it wins (and therefore its opponent loses), otherwise, it loses (and its opponent wins).

Since we model dialogue as a turn-based interaction and we will need to consider joint actions, we introduce the action `listen` corresponding to the empty action.

6 Experimental Setting

Effects of the multi-agent setting are studied here through one special feature of the human-machine dialogue: the uncertainty management due to the dysfunctions of the ASR and the NLU. To promote simple algorithms, we ran our experiments on the zero-sum dialogue game presented above.

On this task, we compare three algorithms: Q-Learning, WoLF-PHC and AGPI-Q. Among those algorithms, only AGPI-Q is proved to converge towards a Nash Equilibrium in a Multi-Agent setting. Q-Learning and WoLF-PHC have however been used as Multi-Agent learning algorithm in a dialogue setting (English and Heeman, 2005; Georgila et al., 2014). Similarly to these papers, experiments will be done using simulation. We will show that, contrarily to AGPI-Q, they do not converge towards the Nash Equilibrium and therefore do not fit to the dialogue problem.

6.1 Modeling ASR and NLU Confidence Estimation

One difficulty while working with Spoken Dialogue Systems is how can a DM deal with uncertainty resulting from ASR and NLU errors and reflected by their Confidence Scores. Those scores are not always a probability. The only assumption made here is that with a score lower (resp. greater) than 0.5, the probability to misunderstand the last utterance is greater (resp. lower) than 0.5. Since dialogues are simulated, the ASR and NLU confidence levels will be modeled the following way.

Each agent owns some fixed Sentence Error Rate (SER_i). With probability $(1 - \text{SER}_i)$, agent i receives each utterance undisrupted, while with probability SER_i, this utterance is misunderstood and replaced by another one.

A $(-\infty, \infty)$ score is then sampled according to a normal distribution centered in -1 for incorrect understanding and +1 for correct understanding. The (0,1) score is obtained by applying the sigmoid function $f(x) = \frac{1}{1+\exp(-x)}$, to the $(-\infty, \infty)$ score.

Since Q-Learning and WoLF-PHC are used in their tabular form, it was necessary to discretize this score. To have states where the agent is almost sure of having understood (or sure of having misunderstood), we discretized by splitting the score around the cut points 0.1, 0.5 and 0.9. By equity concerns, the same discretization was applied for the AGPI-Q algorithm.

6.2 Task Modeling

6.2.1 State Space

Consider two agents i and j. Their secret numbers are respectively m and n. To gather information about m, agent i asks if the secret number m is smaller or greater than some given number k. If agent j answers that m is greater (resp. smaller) than k, it will provides i a lower bound b_i (resp. an upper bound b'_i) on m. Agent i's knowledge on m may be represented by the interval $I_i = [b_i, b'_i]$. The probability of wining by making a `guess` is then given by $p = \frac{1}{b'_i - b_i + 1}$. Progress of agent i in the game may therefore measured by only $c_i = b'_i - b_i + 1$, the cardinal of I_i. At the beginning of the game, one has: $I_i = I_j = [1, 5]$. Since agents have to know the progress of the whole game, they both track c_i and c_j.

To take an action, an agent needs to remember who pronounced the last utterance, what was the

last utterance it heard and to what extent it believes that what it heard was what had been said.

To summarize, agents taking actions make their decision according to the following features: the last utterance, its trust in this utterance, who uttered it, its progress in the game and its opponent's progress. they do not need to track the whole range of possible secret numbers but only the cardinal of these sets. *Dialogue turn*, *last action*, *confidence score*, *cardinal of possible numbers for both agents* are thus the five state features. The state space thus contains $2 * 5 * 4 * 5 * 5 = 1000$ states.

6.2.2 Action Space

Agents are able to make one of the following actions: `ask`, `answer`, `guess`, `confirm` and `listen`. The actions `ask`, `answer` and `guess` need an argument: the number the agent wants to compare to. To learn quicker, we chose not to take a decision about this value. When an agent asks, it asks if the secret number is greater or smaller than the number in the middle of his range (this range is computed by the environment, it is not taken into account in the states). An agent answering says that her secret number is greater or smaller than the number it heard (which may be not the uttered number). An agent guessing proposes randomly a number in his range of possible values.

6.2.3 Reward function

To define the reward function, we consider the maximizing player. It is its turn to play. If it is guessing the right number, it earns $+1$. If it asks for a confirmation, it earns -0.2. Therefore, it is never in its interest to block the dialogue by always asking for a confirmation (in the worst case, *ie* if second agent immediately wins, it earns -1 while if it infinitely blocks the dialogue, it earns $-0.2 \sum_{k=0}^{\infty} (\gamma^2)^k \approx -1.05$ for $\gamma = 0.9$).

6.3 Training of the algorithms

To train Q-Learning and WoLF-PHC, we followed the setup proposed in (Georgila et al., 2014). Both algorithms are trained in self-play by following an ϵ-greedy policy. Training is split into five epochs of 100000 dialogues. The exploration rate is set to 0.95 in the first epoch, 0.8 in the second, 0.5 in the third, 0.3 in the fourth and 0.1 in the fifth.

The parameters δ_L and δ_W of WoLF-PHC are set to $\delta_W = 0.05$ and $\delta_L = 0.2$. The ratio $\delta_L / \delta_W = 4$ assures an aggressive learning when losing.

As a batch RL algorithm, AGPI-Q requires samples. To generate them, we followed the setup proposed in (Pietquin et al., 2011). An optimal (or at least near) policy is first handcrafted. This policy is the following: an agent always `asks` for more information except when it or its opponent have enough information to make the right `guess` with probability 1. When the agent has to answer, it asks to `confirm` if its confidence score is below 0.5.

An ϵ-random policy is then designed. Agents make their decisions according the hand-crafted policy with probability ϵ and pick randomly actions with probability $(1 - \epsilon)$. Tuples (s, a_1, a_2, r, s') are then gathered. We are then assured that the problem space is well-sampled and that there also exists samples giving the successful task completion reward. To ensure convergence, 75000 such dialogues are generated.

To keep the model as parameter-free as possible, CART trees are used as hypothesis space for the regression.

Each algorithm is trained with the following SER values: 0, 0.1, 0.2, 0.3 and 0.4.

6.4 Results

The decision in the game is made on only two points: when is the best moment to end the dialogue with the `guess` action and what is the best way to deal with uncertainty by the use of the `confirm` action. Average duration of dialogues and average number of `confirm` actions are therefore chosen as the feature characterizing the Nash Equilibrium. Both are calculated over 5000 dialogues. Figures 1 and 2 illustrate those results.

Q-Learning dialogues' length decreases gradually with respect to an increasing SER (Figure 1). Figure 2 brings an explanation: Q-Learning agents do not learn to use the `CONFIRM` action. More, dialogue length is even not regular, proving that the algorithm did not converge to a 'stable' policy. Q-Learning is a slow algorithm and therefore, agents do not have enough time to face the non-stationarities of the multi-agent environment. Convergence is thus not possible.

WoLF-PHC does not treat uncertainty too. Its number of `confirm` actions is by far the highest but stays constant. If the SDS asks for confirmation, even when there is no noise, it may be because being disadvantaged, it always loses, and

while losing, its quick learning rate makes its strategy always changing. As previously said, convergence was not guaranteed.

AGPI-Q is then the only algorithm providing robustness against noise. The length of dialogues and the number of `confirm` actions increase both gradually with the SER of the SDS. We are also assured by the theory that in this setting, no improvement is possible.

It is also interesting to note the emergence of non-trivial strategies coming from the interaction between the AGPI-Q agents. For instance, when both agents are almost at the end of the dialogue ($c_i = 2$ for each agent), agents make `guess`. Even if they have very low chances of wining, agents make also `guess` when it is sure that the adversary will win at the next turn.

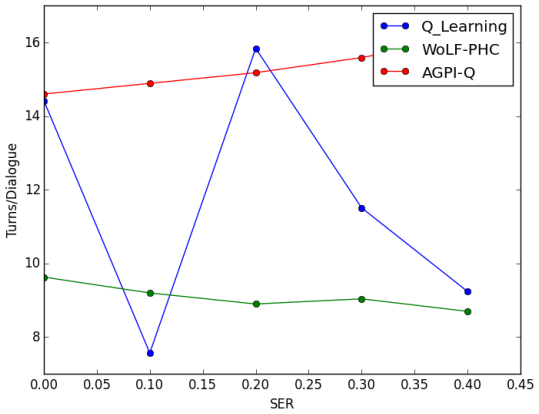

Figure 1: Length of dialogues

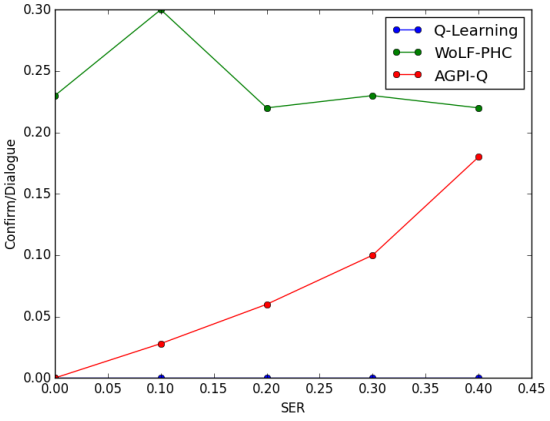

Figure 2: Frequency of the action `CONFIRM`

7 Conclusion: Beyond the Zero-Sum Setting

We provided a rigorous framework for co-learning in Dialogue Systems allowing optimization for both conversants. Its efficiency was shown on a purely adversarial setting under noisy conditions and an extension to situations more general than the purely adversarial setting is now proposed.

7.1 An appointment scheduling problem

The previous model considers only purely competitive scenarios. In this section, it is extended for the General-Sum case. We take as an example the task of scheduling the best appointment between two agents, where conversants have to interact to find an agreement.

Each agent i has its own preferences about a slot in their agenda, they are encoded into some reward function $\mathcal{R}_i$. At each turn, an agent `proposes` some slot k. Next turn, its interlocutor may `propose` another slot or `accept` this one. If it accepts, agent i earns $\mathcal{R}_i(k)$, it gets nothing otherwise. The conversation ends when an agent accepts an offered slot.

Agents, which are not always perfect, can misunderstand the last offer. An action `confirm` is therefore introduced. If an agent thinks that the last offer was on the slot k' instead of the slot k, the outcome may be disastrous. An agent has thus always to find a trade-off between the uncertainty management on the last offer and its impatience, (due to the discount factor γ which penalizes long dialogues).

Here, cooperation is implicit. Conversants are self-centered, they care only on their own value functions, but, since it depends on both actions, or more explicitly the opponent may refuse an offer, they have to take into account the opponent's behavior.

7.2 Future work

In future, using General-Sum algorithms (Prasad et al., 2015), our framework will be applied on those much more complicated dialogue situations where cooperative and competitive phenomenon get mixed up in addition to the noisy conditions encountered in dialogue.

The long-term goal of this work is to use the model on a real data set in order to provide model of real interactions and designing adaptive SDS freeing ourselves from user modeling.

Acknowledgement

This work has been partially funded by the French National Agency for Research (ANR) through the ContInt Project MaRDi (Man-Robot Dialogue) and by the French Ministry for Higher Education and Research (MESR).

References

Natalia Akchurina. 2009. Multiagent reinforcement learning: algorithm converging to nash equilibrium in general-sum discounted stochastic games. In *Proc. of AAMAS*.

Dimitri P. Bertsekas and John Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Grégory Bourguin, Alain Derycke, and Jean-Claude Tarby. 2001. Beyond the interface: Co-evolution inside interactive systems - a proposal founded on activity theory. In *People and Computers XV-Interaction without Frontiers*, pages 297–310. Springer.

Michael Bowling and Manuela Veloso. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250.

Lucian Buşoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172.

Jean Caelen and Anne Xuereb. 2011. Dialogue et théorie des jeux. In *Congrés international SPeD*.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. 2011. User Simulation in Dialogue Systems using Inverse Reinforcement Learning. In *Proc. of Interspeech*.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. 2012a. Behavior Specific User Simulation in Spoken Dialogue Systems. In *Proc. of ITG Conference on Speech Communication*.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. 2012b. Co-adaptation in Spoken Dialogue Systems. In *Proc. of IWSDS*.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902.

Ioannis Efstathiou and Oliver Lemon. 2014. Learning non-cooperative dialogue behaviours. In *Proc. of SIGDIAL*.

Layla El Asri, Romain Laroche, and Olivier Pietquin. 2014. Dinasti : Dialogues with a negotiating appointment setting interface. In *Proc. of LREC*.

Michael S. English and Peter A. Heeman. 2005. Learning mixed initiative dialog strategies by using reinforcement learning on both conversants. In *Proc. of HLT/EMNLP*.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. pages 503–556.

Jerzy Filar and Koos Vrieze. 1996. *Competitive Markov decision processes*. Springer.

Matthieu Geist, Olivier Pietquin, and Gabriel Fricout. 2009. Tracking in reinforcement learning. In *Proc. of ICONIP*.

Kallirroi Georgila, Claire Nelson, and David Traum. 2014. Single-agent vs. multi-agent techniques for concurrent reinforcement learning of negotiation dialogue policies. In *Proc. of ACL*.

Geoffrey J. Gordon. 1999. *Approximate Solutions to Markov Decision Processes*. Ph.D. thesis, Carnegie Melon University.

P. Jean-Jacques Herings and Ronald Peeters. 2000. Stationary equilibria in stochastic games: structure, selection and computation.

Junling Hu and Michael P. Wellman. 2003. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069.

Romain Laroche, Ghislain Putois, and Philippe Bretier. 2010. Optimising a handcrafted dialogue system design. In *Proc. of Interspeech*.

Oliver Lemon and Olivier Pietquin. 2007. Machine learning for spoken dialogue systems. In *Proc. of Interspeech*.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proc. of Eurospeech*.

Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of ICML*.

Michael L. Littman. 2001. Friend-or-foe q-learning in general-sum games. In *Proc. of ICML*.

John Milnor. 1951. Games against nature. Technical report, RAND corporation.

Abraham Neyman and Sylvain Sorin. 2003. *Stochastic games and applications*, volume 570. Springer Science & Business Media.

Martin J. Osborne and Ariel Rubinstein. 1994. *A course in game theory*. MIT press.

Stephen D. Patek and Dimitri P. Bertsekas. 1999. Stochastic shortest path games. *SIAM Journal on Control and Optimization*, 37(3).

Julien Perolat, Bilal Piot, Bruno Scherrer, and Olivier Pietquin. 2015. Approximate dynamic programming for two-player zero-sum markov games. In *Proc. of ICML*.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(01):59–73.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3).

Olivier Pietquin. 2006. Consistent goal-directed user model for realistic man-machine task-oriented spoken dialogue simulation. In *Proc of ICME*.

H.L. Prasad, L.A. Prashanth, and Shalabh Bhatnagar. 2015. Algorithms for nash equilibria in general-sum stochastic games. In *Proc. of AAMAS*.

Martin L. Puterman. 1994. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Proc. of HLT*.

Jost. Schatztnann, Matthew Stuttle, Konrad Weilhammer, and Steve Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *Proc. of ASRU*.

Lloyd Shapley. 1953. Stochastic games. *Proc. of the National Academy of Sciences of the United States of America*, 39(10):1095–1100.

Satinder P. Singh, Michael J. Kearns, Diane J. Litman, and Marilyn A. Walker. 1999. Reinforcement learning for spoken dialogue systems. In *Proc. of NIPS*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning: An introduction*. MIT press.

Christopher Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Steve Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Martin Zinkevich, Amy Greenwald, and Michael L. Littman. 2006. Cyclic equilibria in markov games. In *Proc. of NIPS*.