Reinforcement Learning for Turn-Taking Management in Incremental Spoken Dialogue Systems

Hatim Khouzaimi
Orange Labs, CERI-LIA
hatim.khouzaimi@gmail.com

Romain Laroche
Orange Labs, Châtillon, France
romain.laroche@orange.com

Fabrice Lefèvre
CERI-LIA, Avignon, France
fabrice.lefevre@univ-avignon.fr

Abstract

In this article, reinforcement learning is used to learn an optimal turn-taking strategy for vocal human-machine dialogue. The Orange Labs' Majordomo dialogue system, which allows the users to have conversations within a smart home, has been upgraded to an incremental version. First, a user simulator is built in order to generate a dialogue corpus which thereafter is used to optimise the turn-taking strategy from delayed rewards with the Fitted-Q reinforcement learning algorithm. Real users test and evaluate the new learnt strategy, versus a non-incremental and a handcrafted incremental strategies. The data-driven strategy is shown to significantly improve the task completion ratio and to be preferred by the users according to subjective metrics.

1 Introduction

Building Spoken Dialogue Systems (SDSs) is motivated by the desire to improve human-machine interaction efficiency as well as the will to increase the systems' naturalness and human-likeness. A major sticking point encountered in currently deployed SDSs is the oversimplified rigid turn-taking model which makes dialogues look like walkie-talkie conversations instead of natural interactions: while the user speaks, the system can do nothing but listen and vice versa. The system only starts processing the user's utterance after it is finished, as indicated by a short silence. As it has been shown in several studies, humans behave differently since they process the speaker's utterance as it is spoken, before its end [Tanenhaus *et al.*, 1995] which makes them able to perform a series of complex turn-taking behaviours [Sacks *et al.*, 1974; Khouzaimi *et al.*, 2015b]. Therefore, for nearly two decades, an active research thread has been dedicated to exploring the possibility of replicating such behaviours in SDSs and studying their impact on human-machine interactions. The techniques developed in this frame are referred to as incremental dialogue processing [Schlangen and Skantze, 2011].

An incremental dialogue system processes the dialogue utterance as it is spoken (e.g. every 500ms or after each change in automatic speech recognition result). This gives it the ability to be more reactive when providing answers to the user's requests and also, to be more proactive in the dialogue by interrupting the user in the case of a misunderstanding for example (hence, fixing desynchronisations between speakers more quickly). Moreover, the user is also allowed to barge-in when the system is taking the floor [El Asri *et al.*, 2014]. This has been shown to improve both dialogue efficiency and human-likeness [Aist *et al.*, 2007; Skantze and Schlangen, 2009; Ghigi *et al.*, 2014].

In parallel, since the beginning of the new century, the SDS research field has also been very interested in optimising dialogue strategies directly from data [Levin and Pieraccini, 1997], mostly using reinforcement learning (RL) algorithms [Sutton and Barto, 1998]. These techniques have the advantage to prevent SDSs designers from handcrafting the whole system and setting all the parameters by hand. This approach has been proved to provide better and more robust results with less labour and time resources [Lemon and Pietquin, 2012; Young *et al.*, 2013]. However, these techniques require important amounts of data which are costly to gather in the field of dialogue. To overcome that, user simulation has been studied [Eckert *et al.*, 1997; Schatzmann *et al.*, 2006; Pietquin and Hastie, 2013].

This article contributes with the first work on RL for incremental dialogue to be evaluated with humans on the live dialogue task. Our method consists in learning an optimal turn-taking strategy directly from delayed rewards (received at the end of each independent task), thus requiring no human intervention (like annotations for supervised learning) nor additional assumptions about desired behaviours (e.g. minimising gaps and overlaps). In addition, this provides an important flexibility since the reward function can be set to whatever utility function one may want to optimise. Then, the strategy learnt [Khouzaimi *et al.*, 2015a] in a simulated environment [Khouzaimi *et al.*, 2016] is tested with real users (206 dialogues) in a fully realistic and unbiased situation (supporting all the variability that is generally inherent to dialogue and even environment configuration variability since users performed the experience from their own devices). For that, we use a standard slot-filling task, similar to many vocal agents currently deployed and the subjective evaluation is performed live since the users had to fill a survey at the end of each dialogue.

Section 2 describes the existing studies related to RL or real human experiments for incremental dialogue systems.

Then Section 3 provides the implementation details of both the simulated environment and the real dialogue system and Section 4 introduces the turn-taking strategies that are studied here (handcrafted and data-driven using RL). Finally, Section 5 describes the experiment and the associated results and Section 6 concludes and provides some perspectives left for future work.

2 Related work

Previous studies in the field of incremental dialogue processing can be categorised according to two aspects: whether they use handcrafted, supervised learning or RL based strategies and whether their results have been tested and validated using a real live study.

Firstly, some studies rely on a handcrafted or a supervised learning approach that is evaluated outside of a real dialogue situation. [Aist *et al.*, 2007] shows that incremental processing significantly reduces the dialogue duration while improving the user satisfaction using a handcrafted setup. However, the dialogues collected are not live interactions (pre-recorded in-domain and clearly understandable user utterances) and judges evaluate the system by watching videos of the interaction. A similar evaluation approach is used in [Meena *et al.*, 2013] where a classifier learns the right moment for the system to perform feedbacks and backchannels. Finally, [Zhao *et al.*, 2015] proposes a new classification approach (the rewards are inspired by the way delayed rewards are modeled in RL) to learn the right moment to take the floor which has been evaluated in simulation.

Secondly, a few papers also focus on handcrafted and supervised learning approach but they evaluate them using real live dialogues. A system that is able to incrementally perform acknowledgments (backchannel), clarification requests and repetition (feedback) is described in [Skantze and Schlangen, 2009] (using a number dictation micro-domain). A rule-based turn-taking strategy is used and the results show that human-likeness is improved but no effect has been reported concerning the dialogue efficiency. In [Raux and Eskenazi, 2009], a state machine model is introduced with the aim of minimising gaps and overlaps (principle introduced in [Sacks *et al.*, 1974] while analysing human-human conversations). A handcrafted cost function is used and compared to a fixed silence threshold policy, it is shown in a live dialogue study that the resulting system reduces the dialogue system's response latency with lower false cut-in rates (in the sense that the new strategy less often interrupts the user too early, even though the comparison is not statistically significant). A new model (also handcrafted, with the goal of achieving smooth turn transitions) proposed in [Zhao *et al.*, 2015] significantly improved the task completion on the same task. This task has also been used in [Ghigi *et al.*, 2014] and to our knowledge, this is the first study where the idea of interrupting the user to repair errors have been studied. A handcrafted strategy along with a live dialogue study corpus have made it possible to show a significant improvement in task completion but the dialogues are slightly longer than the baseline. Finally, a very recent study [Paetzel *et al.*, 2015] compares dialogue systems with different degrees of incrementality. While interacting with the system, the users received higher incentives if they manage to accomplish more tasks in a short period of time. As a consequence, incremental processing was very helpful since it allows more reactive dialogues but a certain bias is introduced in comparison with a classic dialogue task since the users do not usually feel the urge to interact with the system as fast as possible. A simple model is used for turn-taking management by the system and it is based on two thresholds: an Natural Language Understanding (NLU) confidence threshold above which the system decides to speak in order to accomplish the task and a time threshold above which it decides to abandon the current task and move to the next one (values set using a previously collected corpus).

Finally, as far as RL studies are concerned, a few studies used this framework in order to optimise turn-taking according to different criteria. The existing work in this direction is evaluated using simulation and offline techniques only. [Jonsdottir *et al.*, 2008] introduced a new framework where the system considers only prosodic features (neglecting the meaning of what is being said) and learning smooth turn-taking in an autonomous way. For training, the system interacted with itself and the resulting conversation was compared to real dialogues separately. On the other hand, [Selfridge and Heeman, 2010] proposed a model where the dialogue participant has the most important information to say (in order to move the dialogue forward) takes the floor. In a simulated environment, a RL strategy is shown to reduce the dialogue duration and to improve the task completion. In [Lu *et al.*, 2011], a POMDP-based approach is used in a simple setup where an episode consists on the user speaking then being interrupted by the system that completes her utterance (after trying to guess the remaining part). Only 50 possible utterances are considered. [Dethlefs *et al.*, 2012] uses Hierarchical RL in order to learn both what the incremental dialogue system should say and when it should say it. A reward (positive or negative) is received during the dialogue depending on a few events (task completed, misunderstanding, turn-taking at the right/wrong time given the notion of *information density* introduced in the paper...). A few judges are given utterances where they have three alternatives for barge-in location (they are supposed to select the one where they think a barge-in should be appropriate). One of these three options corresponds to the location where the learnt strategy would barge-in and the other two correspond to two baselines. The users agreed more with the learnt strategy. Another approach based on Inverse RL techniques is introduced in [Kim and Banchs, 2014]. Using a human dialogue corpus, the objective is to learn a policy that imitates it best. The obtained strategy is assessed both in simulation and offline, using a test corpus.

The objective of this paper is to propose a new turn-taking optimisation technique using RL which is thoroughly evaluated with real users in a live dialogue setup. Compared to previous approaches, the system adapts its strategy with no human intervention (unlike supervised learning where important annotations are required) and no assumptions about what would suit the users best (like the urge to minimise gaps and overlaps for example).

3 Implementation

3.1 Architecture overview

Figure 1 depicts the overall architecture of the incremental dialogue system [Khouzaimi *et al.*, 2014].

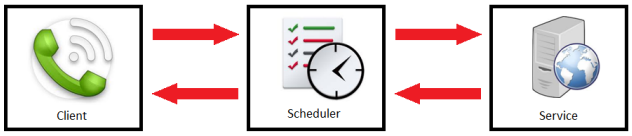

Figure 1: The Scheduler: turn-taking manager module

The Client contains the Automatic Speech Recognition (ASR) and the Text-to-Speech (TTS) modules, hence playing the role of a direct interface that the user can interact with. The Service is a traditional non-incremental dialogue system. It is able to process the dialogue request and to formulate an answer. Therefore, it contains the NLU, the Dialogue Manager (DM) and the Natural Language Generation (NLG) modules. The Scheduler on the other hand manages turn-taking decisions. It receives the ASR results incrementally as the user speaks and then decides whether to take the floor at each new increment or not.

3.2 Majordomo Service

The Service implemented is a Majordomo helping the user to schedule domestic tasks, like laundry, heating etc. The tenant can say, for example: *Can you please turn on the heating on March 3rd from 9pm until 11pm* (dialogues are in French but they are translated here into English for clarity). She can also move a pre-scheduled task to a new date and time or delete it. This is a slot-filling task where four slots are manipulated: the action type (ADD, MODIFY or DELETE), the task (laundry, hoover etc.) the date and the time window. A mixed initiative strategy (non-incremental baseline) is used for the dialogue management in order to gather all the slots: first the user formulates a complete request in natural language and if there are still missing information slots, the system asks for them one by one like in the following example:

USER: Can you please turn on the heating on <noise> from 9pm until 11pm?
SYSTEM: Please specify a date.
USER: I said March 3rd.
SYSTEM: Ok. So you want to turn on the heating on March 3rd from 9pm until 11pm. Is that right?

Moreover, some of the domestic tasks are incompatible and they cannot be scheduled at the same time (for example, mowing the lawn and watering it). This constraint generates more or less complicated dialogues where the user should move or delete conflicting events before being able to accomplish his initial task.

3.3 The Scheduler

The user's utterance is communicated to the Scheduler as it is spoken, therefore, a user's turn is divided into many smaller *micro-turns*. At each micro-turn, a new updated ASR output (corresponding to the whole partial sentence pronounced by the user so far, and not only what has been pronounced during the micro-turn) is sent by the Client to the Scheduler which in turns sends this message to the service and stores the response.

The role of the Scheduler is to decide whether to retrieve the system's response to the Client during the current micro-turn, which is directly communicated to the TTS resulting in the system taking the floor or whether to do nothing and wait for the user to complete her request in the next few micro-turns (in which case the Service's response is discarded). This decision is what is called a *turn-taking decision*.

In addition to the ability to transform a non-incremental Service into an incremental dialogue system, another advantage of using a multi-layer architecture is to separate the turn-taking management part from the traditional dialogue management. As a result, modifying the Scheduler only makes it possible to test several dialogue strategies. In the following, the handcrafted and the data-driven strategies are implemented inside this module.

3.4 Input/ouput Client

For the ASR task, Google's solution has been chosen since it is a state-of-the-art off-the-shelf option and it provides partial results (the moments when they are retrieved are determined by the solution). Other alternatives like Kaldi [Povey *et al.*, 2011] offer the possibility to have a more customised ASR which is more adapted to the task at hand, however, an important amount of work is required in order to prepare it (acoustic models, language models etc.).

We also use Google's TTS solution since, along with the ASR, they are easy to embed in a web client. While the user speaks, Google ASR takes charge of sending the incremental results to the Scheduler and when the latter decides to take the floor, the corresponding message is retrieved for the TTS to pronounce. During the first two seconds of a system's utterance, the ASR in disabled for it to be able to barge-in. Otherwise, the user might take the floor back right after the system tries to speak (probably without noticing the TTS activation).

3.5 User Simulator

In the first part of this study, the Client is replaced with a User Simulator in order to generate an important amount of dialogues that are fed to our RL algorithm. In this section, we rapidly describe the functioning of this simulated environment. For more details, the reader can check [Khouzaimi *et al.*, 2016].

The User Simulator is composed of five modules: the Intent Manager, the NLG, the Verbosity Manager, the NLU and the Patience Manager. The NLU module transforms the Majordomo's sentences into concepts understandable by the Intent Manager and the Patience Manager ends the dialogue whenever the dialogue gets too long (given a random but reasonable threshold). The Intent Manager is given a list of domestic tasks that it is supposed to schedule during the interaction with the Majordomo. It computes the next user dialogue act at each new dialogue turn using an agenda-based approach [Schatzmann *et al.*, 2007]. This dialogue act is then sent to

the NLG which generates a basic utterance like *Cancel the heating task*. Then, the Verbosity Manager either makes it more realistic by adding a prefix and a suffix, for example *I would like to cancel the heating task please*, either replaces it with an off-domain sentence or repeats the same information twice (often happens in real dialogue [Ghigi *et al.*, 2014]).

The Verbosity Manager's output is then fed on a word by word fashion to the ASR Output Simulator module that is in charge of simulating noise and ASR imperfections (hence, a new word pronounced by the user is considered as a new micro-turn here). The Word Error Rate (WER) is provided as a parameter in order to simulate noise and an N-Best containing several recognition hypotheses with the associated confidence scores is generated. Here, in order to keep things simple, only the best hypothesis is sent to the Scheduler.

Finally, timing is estimated in simulation by considering a speech rate of 200 words per minute [Yuan *et al.*, 2006] and by adding one second of silence during a system to user floor transition (provided that no barge-in is involved) and two seconds the other way around.

4 Turn-taking strategies

4.1 Handcrafted strategy

Based on a previous proposed turn-taking taxonomy study in human conversations [Khouzaimi *et al.*, 2015b], a few turn-taking behaviours have been implemented in the Scheduler. Firstly, if the user talks for too long without delivering any understandable and useful partial information, the Scheduler decides to barge-in to report a misunderstanding. An empirical fixed time threshold is fixed for each type of request: 1.5 seconds (5 words) for open questions (where all the slots should be provided in one dialogue turn), 0.6 seconds (2 words) for yes/no questions, 0.9 seconds (3 words) for date questions, 2.7 seconds (9 words) for time window questions and 1.5 seconds (5 words) for domestic task name questions.

Secondly, if the system detects that the user tries to schedule a domestic task which is in conflict with a pre-scheduled one, it barges-in to report the problem. It does the same when the user tries to move or delete a non pre-scheduled task. Also, when all the information needed by the user is provided, the system delivers its responses whether the user stopped talking or not. While the user's request is spoken, early ASR results are not necessarily prefixes of later ones since the ASR result might change overtime when more information is provided. This phenomenon is referred to as ASR instability [Selfridge *et al.*, 2011]. Moreover, earlier words in a partial request ASR result are less likely to change than later ones. [McGraw and Gruenstein, 2012] shows that a word spoken more than 0.6 seconds ago (2 words given a speech rate of 200 words) have a 90% chance of staying unchanged. Therefore, when detecting a reason to barge-in in a user's partial utterance, the Scheduler waits of 2 additional words and if this reason persists, it decides to barge-in.

4.2 Reinforcement learning strategy

Background

Reinforcement learning [Sutton and Barto, 1998] is a machine learning framework where an agent learns to take decisions in a certain environment in order to maximise a reward function through trials and errors. This agent is generally modeled as a Markov Decision Process (MDP) which is a tuple $(\mathcal{S}, \mathcal{A}, \mathscr{T}, \mathscr{R}, \gamma)$ where $\mathcal{S}$ is the state space (all the states the agent can be in), $\mathcal{A}$ is the action space (all the actions the agent can perform), $\mathscr{T}$ the transition model (the transition distribution $\mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$ for every $(s, a, s') \in \mathcal{S}\text{x}\mathcal{A}\text{x}\mathcal{S}$ where s_t and a_t are respectively the agent's state and the action it decides to take at time t), $\mathscr{R}$ the reward model (the immediate reward distribution $\mathbb{P}(r_t | s_t = s, a_t = a, s_{t+1} = s')$ for every $(s, a, s') \in \mathcal{S}\text{x}\mathcal{A}\text{x}\mathcal{S}$). $\gamma \in [0, 1[$ is a discount factor penalising late reward. A deterministic policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ defines the behaviour of the agent. The goal of RL is to find a policy that maximises the expected long term return defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. Such a policy is denoted π^*. For each $(s, a) \in \mathcal{S}\text{x}\mathcal{A}$, the quantity $Q^\pi(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$ defines what is known as the Q-function (the expected long return for being at state s, taking action a and following policy π afterwards). In some cases, it is possible to encode the Q-function in a tabular fashion where all the values corresponding to every couple $(s, a) \in \mathcal{S}\text{x}\mathcal{A}$ are kept separately but in most cases, the state space is continuous which makes it impossible to proceed as such. As a consequence, a parametric representation $\tilde{Q}$ is used as an approximation of the real Q-function. Here, a linear representation is used where the state s is represented through a vector of features $\phi(s)$ such that, by specifying a parameter vector $\theta(a)$ for each action a, the $\tilde{Q}$-function can be written $\tilde{Q}^\pi(s, a) = \theta(a).\phi(s)$.

$\tilde{Q}^* = \tilde{Q}^{\pi^*}$ verifies the following equation for each $(s, a) \in \mathcal{S}\text{x}\mathcal{A}$ (called the Bellman optimality equation):

$$Q^*(s_t, a_t) = \mathbb{E}[r_t + \gamma \max_{a \in \mathcal{A}} Q^*(s_{t+1}, a)] \quad (1)$$

With the linear approximation, it becomes $\tilde{Q}^* = \Pi T \tilde{Q}^*$, where Π is the projection into the space of approximate value functions, and T is the Bellman operator. It turns out that ΠT is a contraction. As a consequence, given data extracted from the agent's experience while interacting with the environment, it is possible to derive an iterative method to converge towards $\tilde{Q}^*$ (Banach theorem) which leads to the Fitted-Q algorithm [Richard Bellman, 1959] (batch algorithm, successfully applied to traditional dialogue management in [Chandramohan *et al.*, 2010]). More precisely, given a set of N transitions and the associated rewards (s_j, a_j, s'_j, r_j), for each action $a \in \mathcal{A}$ ($\theta_i(a)$ being the i^{th} component of the vector $\theta(a)$):

$$\theta_i(a) = \left(\sum_{j=1}^{N} \phi(s_j)^T \phi(s_j) \right)^{-1} \sum_{j=1}^{N} \phi(s_j) \left(r_j + \gamma \max_{a' \in \mathcal{A}} \theta_{i-1}(a')^T \phi(s'_j) \right) \quad (2)$$

The iterative process stops when $|\theta_i(a) - \theta_{i-1}(a)| \leq \xi$ (here, $\xi = 0.01$). The greedy policy is then directly deduced from the $\tilde{Q}$-function: $\forall s \in \mathcal{S}$, $\pi(s) = \arg\max_{a \in A} \tilde{Q}(s, a)$.

Model

At each new micro-turn, the Scheduler receives a new ASR input, asks the Service for a response and based on these elements, it decides whether to do nothing and wait for more ASR inputs (or a silence) or whether to take the floor right away. Therefore, the Scheduler is cast as an MDP with a binary action space $\mathcal{A} = \{\texttt{WAIT}, \texttt{SPEAK}\}$ and where an episode corresponds to a part of dialogue where the user tries to perform one single independent task (scheduling, modifying or deleting a domestic task). At the end of each episode, the system receives a reward equals to $150TC - \Delta_t$ where Δ_t is the total episode duration in seconds and TC equals 1 if the user successfully performed the task and 0 otherwise. Since a linear representation of the Q-function is used, a state representation $\phi(s)$ has to be chosen. Here, the following elements are used[1]:

- `SYSTEM_REQ`: The information that the system is waiting for at the current time. It can be a domestic task, a date, a time window, a confirmation, or the system can wait for the response to an open question where all the slots should be specified.
- `LAST_INCR_RESP`: The last response provided by the Service (recall that at each new micro-turn, the Scheduler sends the ASR result corresponding to the last partial utterance from the user and stores the corresponding response).
- `LAST_CHANGE_AGE`: The time elapsed since `LAST_INCR_RESP` last change (i.e. since the Service changed its mind about the response to provide for the last time).
- `TIME`: Duration of the current episode. Embedding time in the state space is necessary to preserve the Markov property in the MDP.

The first features in the vector $\phi(s)$ corresponding to state s are Kronecker variables $\delta_i(s)$ encoding the $n_c = 81$ possible combinations between `SYSTEM_REQ` and `LAST_INCR_RESP` values ($\delta_i(s)$ equals 1 if the i^{th} combination is realised and 0 otherwise). Then `NB_USER_WORDS` is represented through a set of 3 Radial Basis Functions[2] (RBFs) defined as follows ($i \in \{1, 2, 3\}$) :

$$\phi_i^{age}(s) = \exp\left(\frac{(\texttt{LAST\_CHANGE\_AGE} - \mu_i)^2}{2\sigma_i^2}\right) \quad (3)$$

where $\mu_1 = 0$, $\mu_2 = 5$, $\mu_3 = 10$, $\sigma_1 = 2$, $\sigma_2 = 3$ and $\sigma_3 = 3$. Since all these features are between 0 and 1 and in order to improve the model interpretability (for all the $\theta_i(a)$ to have the same order of magnitude), `TIME` is normalised in order to fit in this interval as well (the result is called T). The feature vector is built as follows for each state $s \in \mathcal{S}$: $\phi(s) = [\delta_1(s),...,\delta_{n_c}(s), \phi_1^{age}(s), \phi_2^{age}(s), \phi_3^{age}(s), T]^T$.

[1]The ASR score would also be an interesting feature to include, however, Google ASR does not currently provide intermediate confidence scores (it computes the score at the end of the sentence only).

[2]The means and standard deviations were calibrated empirically.

Learning simulation results

This model was trained in simulation (on 3 different dialogue scenarios with $WER = 0.15$ and $\gamma = 0.99$) and after each 500 new episodes, Fitted-Q was run on the whole collected batch in order to update the $\theta_i(a)$ parameters (Equation 3). The Scheduler learns for 2500 episodes. During the first 500 episodes, the Scheduler randomly picks the action `WAIT` 90% of the time and the action `SPEAK` with a 10% probability (if no bias is introduced in favour of the `WAIT` action, the Scheduler interrupts the user too often which hurts the exploration process) and between episodes 500 and 2500, the Scheduler is greedy with a 0.9 probability and chooses randomly between the two actions the rest of the time (this time, they are picked uniformly). Figure 2 depicts the corresponding learning curve (mean over 50 learning samples, *RL* is the name of the RL strategy) with the non-incremental strategy (*None*) and the Handcrafted one (*Handcrafted*) taken as baselines. An additional set of 500 episodes is added for evaluation (100% greedy policy which explains the jump at the end of the curve).

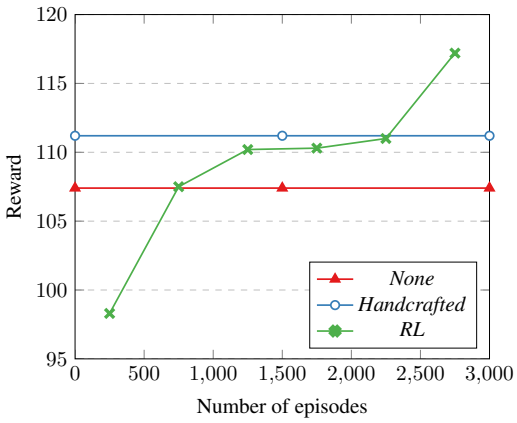

Figure 2: Learning curve (0-500: pure exploration, 500-2500: exploration/exploitation, 2500-3000: pure exploitation)

5 Live study

5.1 Experiment

With the participation of 47 volunteers, 206 dialogues were gathered (*None*: 65, *Handcrafted*: 65, *RL*: 76). The dialogue duration as well as the dialogue duration per task were directly extracted from the dialogue logs. Moreover, the list of the tasks that Majordomo has scheduled is also logged and compared to a reference list in order to determine whether the user completed the task or not. In addition, at the end of each dialogue, users were asked to fill a survey where the following Key Performance Indicators (KPIs) are collected (they are assessed using a Likert scale between 1 and 6 except for *Potential use* where it goes from 1 to 4):

- Reactivity: Is the system reactive?
- Reactivity quality: How do you judge the system's reactivity?

- Human-likeness: Does the system act like a human?
- Efficiency: How do you judge the system's efficiency?
- Overall quality: How do you judge the overall quality of the dialogue?
- Potential use: Considering the last dialogue, would you use the Majordomo at home?

5.2 Results

Table 1 provides the mean values corresponding to the different objective and subjective metrics for the three compared strategies[3]. The mean dialogue duration only slightly improved when using incremental strategies, with no statistical significance. Interestingly, this small gain in dialogue duration leads to a critical task completion improvement.

The *RL* strategy significantly improves the task completion ratio (15% gain with $p = 0.03$). A visible improvement is also shown over the handcrafted baseline even though the p-value is above 0.05 ($p = 0.065$). Finally the handcrafted strategy seems to improve the non-incremental task completion ratio by 3% only in a non-significant way ($p = 0.36$). The Majordomo task implies a certain cognitive load and requires the user to be focused in order to keep track of the succession of tasks that she has to accomplish (not forgetting the initial objective, listening carefully to the conflicts reported by the system in order to act on them...). It appears here that allowing the system to interrupt users in the right moment makes it more look like it is more engaged in the dialogue which helps increasing their focus and making them accomplish their goal (users feel in turn more engaged).

The subjective metrics are all in favour of the *RL* strategy, except for the reactivity quality, but the only significant bias appears when compared with *None* in terms of reactivity ($p = 0.048$). This is due to the fact that subjective metrics are subject to an additional inter-user variability which makes the distributions difficult to separate on a middle-size corpus like the one used here. However, the general trend shows that the *RL* strategy was preferred by the users over the non-incremental and the handcrafted baselines.

Category	KPI	*None*	*Handcrafted*	*RL*
Objective	Duration (sec)	94.7	**89.6**	90.6
	Task completion	0.60	0.63	**0.75**
Subjective	Reactivity	4.31	4.57	**4.62**
	Reactivity quality	**4.38**	4.25	4.36
	Human-likeness	3.63	3.66	**3.74**
	Efficiency	4.22	4.20	**4.36**
	Global quality	4.06	4.18	**4.20**
	Potential use	2.66	2.68	**2.82**

Table 1: Global dialogue evaluation metrics

Incremental processing intrinsically involves a trade-off between reactivity and the risk of interrupting the user too early, referred to as *false cut-in* (FC) in [Raux and Eskenazi, 2012]. To estimate our system's reactivity, each time it took the floor (number of transitions: 549 for *None*, 542 for *Handcrafted* and 727 for *RL*), the latency between the moment when the Scheduler received the last ASR input and the moment it decides to take the floor is measured. This is only an approximation of the real latency defined as the difference between the moment when the voice activity detection signal drops (if no system barge-in) and the moment that the TTS starts. Still, this is a good proxy for strategy comparison.

Table 2 reports the mean values of the three strategies (significant differences: $p < 0.000001$). The handcrafted strategy reduces non-incremental latency by 250 ms whereas the *RL* one is 1 second faster (latency of *None* strategy divided by three). Nevertheless, improving latency does not necessarily translate in better dialogues since the system can be too aggressive hence hurting the interaction quality. To assess that, all the times when the Scheduler picked the action `SPEAK` (*Handcrafted*: 99, *RL*: 456) where manually annotated for being a good barge-in or a FC. The FC rates are also depicted in the Table 2 and it comes out that the handcrafted strategy performs poorly since one third of its barge-in decisions are FC. This is significantly improved by the *RL* strategy ($p < 0.000001$) where this rate is reduced to 6.8%. The mean number of FCs per dialogue is 0.47 for *Handcrafted* and 0.41 for *RL*. As a conclusion, *Handcrafted* takes less interruption risk and when it does, it is poorly managed. On the other hand, the *RL* strategy barges-in more often and most often at the right time which translates into a more fluid dialogue where turns are exchanged smoothly (more human-like).

KPI	*None*	*Handcrafted*	*RL*
Latency (ms)	1545 ± 61	1303 ± 78	$\mathbf{588 \pm 59}$
FC ratio	No barge-in	0.31 ± 0.091	$\mathbf{0.068 \pm 0.023}$

Table 2: Local dialogue evaluation metrics

6 Conclusion and future work

A new data-driven turn-taking strategy using RL has been presented in this paper. To our knowledge, it is the first turn-taking strategy learnt directly from delayed rewards (delivered only at the end of a task) and evaluated with real users in a live experiment. Compared to a non-incremental baseline as well as a rule-based handcrafted strategy, it is shown to significantly improve the task completion ratio and to be perceived as being more reactive. The overall user's judgment is also in favour of this strategy. A more local analysis shows that this new strategy significantly reduces the system's response latency and compared to the handcrafted baseline, it rarely interrupts the user too early.

The strategy is learnt in simulation before being tested in front of real users. The next step that we plan to take is to use the same methodology while learning on-line while interacting with real users. We believe that there is still room for improvement since user simulation techniques are far from perfect. Moreover, we plan to use the data gathered from this experiment in order to improve our simulator in turn.

[3]To assess statistical significance, the Welsh t-test is used for all the metrics (since it is more powerful than non-parametric tests and since we have enough data to assume that the mean follows a normal distribution). For task completion, the more adapted binomial proportions test has also been used for verification and it led to very similar p-values.

References

[Aist *et al.*, 2007] Gregory Aist, James Allen, Ellen Campana, Carlos Gomez Gallo, Scott Stoness, Mary Swift, and Michael K. Tanenhaus. Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods. In *Decalog*, 2007.

[Chandramohan *et al.*, 2010] Senthilkumar Chandramohan, Matthieu Geist, and Olivier Pietquin. Optimizing spoken dialogue management with fitted value iteration. In *Interspeech*, 2010.

[Dethlefs *et al.*, 2012] Nina Dethlefs, Helen Wright Hastie, Verena Rieser, and Oliver Lemon. Optimising incremental dialogue decisions using information density for interactive systems. In *EMNLP-CoNLL*, 2012.

[Eckert *et al.*, 1997] W. Eckert, E. Levin, and R. Pieraccini. User modeling for spoken dialogue system evaluation. In *ASRU*, 1997.

[El Asri *et al.*, 2014] Layla El Asri, Remi Lemonnier, Romain Laroche, Olivier Pietquin, and Hatim Khouzaimi. NASTIA: Negotiating Appointment Setting Interface. In *LREC*, 2014.

[Ghigi *et al.*, 2014] Fabrizio Ghigi, Maxine Eskenazi, M Ines Torres, and Sungjin Lee. Incremental dialog processing in a task-oriented dialog. In *Interspeech*, 2014.

[Jonsdottir *et al.*, 2008] Gudny Ragna Jonsdottir, Kristinn R. Thorisson, and Eric Nivel. Learning smooth, human-like turntaking in realtime dialogue. In *IVA*, 2008.

[Khouzaimi *et al.*, 2014] Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. An easy method to make dialogue systems incremental. In *SIGDIAL*, 2014.

[Khouzaimi *et al.*, 2015a] Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. Optimising turn-taking strategies with reinforcement learning. In *SIGDIAL*, 2015.

[Khouzaimi *et al.*, 2015b] Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. Turn-taking phenomena in incremental dialogue systems. In *EMNLP*, 2015.

[Khouzaimi *et al.*, 2016] Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. Incremental human-machine dialogue simulation. In *IWSDS*, 2016.

[Kim and Banchs, 2014] Seokhwan Kim and Rafael E. Banchs. Sequential labeling for tracking dynamic dialog states. In *SIGDIAL*, 2014.

[Lemon and Pietquin, 2012] Oliver Lemon and Olivier Pietquin. *Data-Driven Methods for Adaptive Spoken Dialogue Systems*. Springer Publishing Company, Incorporated, 2012.

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

[Lu *et al.*, 2011] Di Lu, Takuya Nishimoto, and Nobuaki Minematsu. Decision of response timing for incremental speech recognition with reinforcement learning. In *ASRU*, 2011.

[McGraw and Gruenstein, 2012] Ian McGraw and Alexander Gruenstein. Estimating word-stability during incremental speech recognition. In *Interspeech*, 2012.

[Meena *et al.*, 2013] Raveesh Meena, Gabriel Skantze, and Joakim Gustafson. A data-driven model for timing feedback in a map task dialogue system. In *SIGDIAL*, 2013.

[Paetzel *et al.*, 2015] Maike Paetzel, Ramesh Manuvinakurike, and David DeVault. "so, which one is it?" the effect of alternative incremental architectures in a high-performance game-playing agent. In *SIGDIAL*, 2015.

[Pietquin and Hastie, 2013] Olivier Pietquin and Helen Hastie. A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*, 28, 2013.

[Povey *et al.*, 2011] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE ASRU Workshop*, 2011.

[Raux and Eskenazi, 2009] Antoine Raux and Maxine Eskenazi. A finite-state turn-taking model for spoken dialog systems. In *NAACL*, 2009.

[Raux and Eskenazi, 2012] Antoine Raux and Maxine Eskenazi. Optimizing the turn-taking behavior of task-oriented spoken dialog systems. *ACM Transactions on Speech and Language Processing*, 9(1):1:1–1:23, 2012.

[Richard Bellman, 1959] Stuart Dreyfus Richard Bellman. Functional approximations and dynamic programming. *Mathematical Tables and Other Aids to Computation*, 13:247–251, 1959.

[Sacks *et al.*, 1974] Harvey Sacks, Emanuel A. Schegloff, and Gail Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50:696–735, 1974.

[Schatzmann *et al.*, 2006] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21, 2006.

[Schatzmann *et al.*, 2007] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL*, 2007.

[Schlangen and Skantze, 2011] David Schlangen and Gabriel Skantze. A general, abstract model of incremental dialogue processing. *Dialogue and Discourse*, 2:83–111, 2011.

[Selfridge and Heeman, 2010] Ethan Selfridge and Peter A. Heeman. Importance-driven turn-bidding for spoken dialogue systems. In *ACL*, pages 177–185, 2010.

[Selfridge *et al.*, 2011] Ethan O. Selfridge, Iker Arizmendi, Peter A. Heeman, and Jason D. Williams. Stability and accuracy in incremental speech recognition. In *SIGDIAL*, 2011.

[Skantze and Schlangen, 2009] Gabriel Skantze and David Schlangen. Incremental dialogue processing in a micro-domain. In *ACL*, 2009.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, An Introduction*. The MIT Press, Cambridge, Massachusetts, London, England, 1998.

[Tanenhaus *et al.*, 1995] Michael K. Tanenhaus, Michael J. Spivey-Knowlton, Kathleen M. Eberhard, and Julie C. Sedivy. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634, 1995.

[Young *et al.*, 2013] Stephanie Young, Milica Gasic, Blaise Thomson, and John D Williams. Pomdp-based statistical spoken dialog systems: A review. *IEEE*, 101(5):1160–1179, 2013.

[Yuan *et al.*, 2006] Jiahong Yuan, Mark Liberman, and Christopher Cieri. Towards an integrated understanding of speaking rate in conversation. In *Interspeech*, 2006.

[Zhao *et al.*, 2015] Tiancheng Zhao, Alan W Black, and Maxine Eskenazi. An incremental turn-taking model with active system barge-in for spoken dialog systems. In *SIGDIAL*, 2015.