Optimising Turn-Taking Strategies With Reinforcement Learning

Conference Paper · September 2015

DOI: 10.18653/v1/W15-4643

Hatim Khouzaimi
Orange Labs / Laboratoire Informatique d'Avi…
12 PUBLICATIONS **42** CITATIONS

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

Optimising Turn-Taking Strategies With Reinforcement Learning

Hatim Khouzaimi
Orange Labs
LIA-CERI
France
hatim.khouzaimi@orange.com

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

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

Abstract

In this paper, reinforcement learning (RL) is used to learn an efficient turn-taking management model in a simulated slot-filling task with the objective of minimising the dialogue duration and maximising the completion task ratio. Turn-taking decisions are handled in a separate new module, the Scheduler. Unlike most dialogue systems, a dialogue turn is split into micro-turns and the Scheduler makes a decision for each one of them. A Fitted Value Iteration algorithm, Fitted-Q, with a linear state representation is used for learning the state to action policy. Comparison between a non-incremental and an incremental handcrafted strategies, taken as baselines, and an incremental RL-based strategy, shows the latter to be significantly more efficient, especially in noisy environments.

1 Introduction

Most dialogue systems use a simple turn-taking model: the user speaks and when she finishes her utterance, the system detects a long enough silence and speaks afterwards. Quite often the latter cannot be interrupted neither. On the contrary, incremental dialogue systems are able to understand the user's utterance on the fly thus enabling a richer set of turn-taking behaviours. They can interrupt the user and quickly report a problem. They can be interrupted as well. In this paper, we explore the extent to which such capacity can improve the overall dialogue efficiency. Reinforcement learning (Sutton and Barto, 1998) is used to find optimal strategies.

Human beings use a rich set of incremental behaviours which help them recover from errors efficiently. As soon as a conversation participant detects a problem, she is able to interrupt the speaker so that he can correct his utterance or repeat a part of it for example. In this work, we implement in an expert handcrafted way 3 turn-taking phenomena amongst those classified in the taxonomy proposed in (Khouzaimi et al., 2015a). The resulting strategy is shown to achieve better performance than a non-incremental handcrafted strategy. Then, it is compared to an automatically learned incremental strategy and the latter is shown to achieve even better results.

Machine learning algorithms often need important sets of data in order to converge. In the field of dialogue systems, gathering data is expensive and as a consequence, researchers use simulated users for learning (Eckert et al., 1997; Chandramohan et al., 2011; Pietquin and Hastie, 2013). To run the experiments in this work, a simulated user interacts with a service that manages a personal agenda (Khouzaimi et al., 2015a).

In our work, the turn-taking task is separated from the common dialogue management one and it is handled by a separated module called the *Scheduler* (Khouzaimi et al., 2014). A considerable asset of this architecture is that it can just be added to the agenda service in order to make it incremental. Two versions of this module have been developed: the first one embeds the handcrafted strategy and the second one uses reinforcement learning to optimise turn-taking decisions with respect to objective criteria. Our goal is to improve the dialogue efficiency, therefore, as evaluation criteria and in order to design a reward function, dialogue duration and task completion are used. Fitted-Q (a Fitted Value Iteration algorithm) was used and we show that the optimal policy is quickly learned and that it outperforms both the non-incremental and the handcrafted strategies. These three strategies are then compared under different noise conditions and the automatically learned strategy is proven to be the most robust to high levels of noise.

Section 2 presents some related work and Sec-

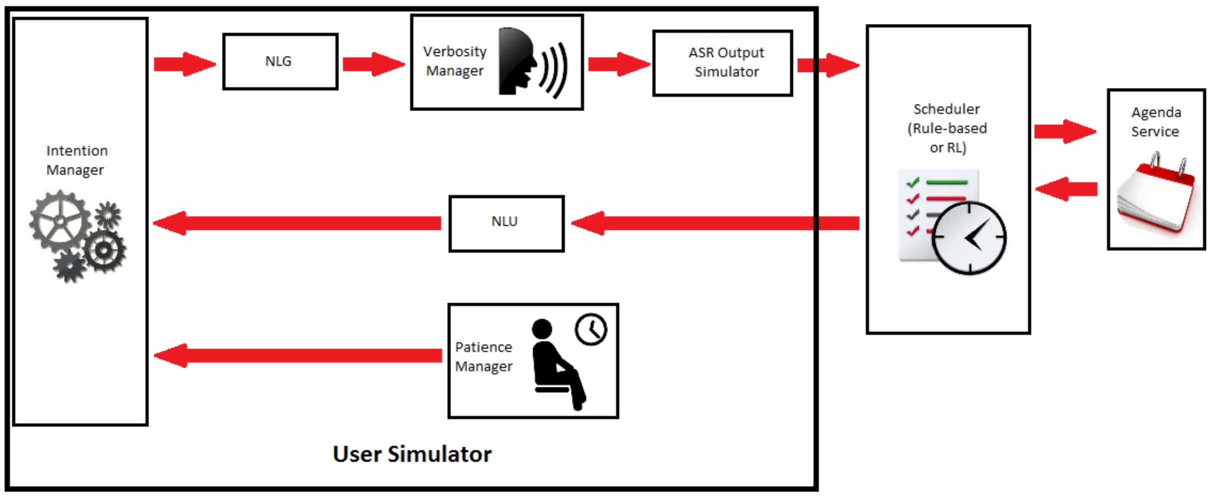

Figure 1: Simulated environment architecture

tion 3 describes the simulated environment used for the experiments. Then Section 4 describes the handcrafted turn-taking model as well as the RL one. Section 5 presents the experimentation and the results and finally, Section 6 gives some concluding remarks.

2 Related work

The idea of interrupting the user in order to improve the dialogue efficiency in terms of dialogue duration and task completion is tackled in (Ghigi et al., 2014). A corpus study shows that the users' utterances often go off-domain or contain the same piece of information several times. By detecting this kind of sentences and interrupting the user to report the problem promptly, the dialogue is more efficient and users tend to conform to the words and expressions that are known by the system. Only handcrafted settings are explored.

An approach based on Hierarchical Reinforcement Learning is presented in (Dethlefs et al., 2012; Hastie et al., 2013). An efficiency reward is used to optimise the Information Presentation strategy (common dialogue management task) whereas another reward based on Information Density is used for the barge-in and backchannel tasks. In our work, the efficiency reward is directly applied to turn-taking management.

A research branch in incremental dialogue focuses on the following principle laid in (Sacks et al., 1974): *Participants in a conversation attempt to minimize gaps and overlaps*.(Jonsdottir et al., 2008) uses a reinforcement learning approach based on this principle in order to achieve smooth turn-taking (only prosodic features are considered) whereas (Raux and Eskenazi, 2008; Raux and Eskenazi, 2012) proposes a classification method where the costs for silences and overlaps are handcrafted. Like the majority of contributions in the field of incremental dialogue, the main focus here is smooth turn-taking rather than improving the general dialogue efficiency.

In order to mimic human turn-taking capabilities, in (Kim et al., 2014) Inverse Reinforcement Learning has been applied to a system that can perform three turn taking actions: *speak*, *silent* and *overlap*. The main focus here is also end of utterance detection and smooth turn-taking.

In (DeVault et al., 2011), the ability of incremental dialogue systems to guess the remaining part of a user's utterance before its end is explored. (Lu et al., 2011) applies reinforcement learning to explore the tradeoff between the risk of error relative to a barge-in due to an early guess and the lack of reactivity in the case of late system responses.

Finally, reinforcement learning is also applied in (Selfridge and Heeman, 2010) in the case of mixed initiative dialogue systems. However the paper does not tackle the problem of barge-in management but initial turn-taking (who takes the floor first): the dialogue participant that has the most important thing to say to make progress in the dialogue takes the floor first.

3 Simulated environment

To learn the turn-taking policy, a simulated environment has been developed. Figure 1 gives an overview of its architecture. The six modules on the left constitute the user simulator: the Intention Manager, the Natural Language Generator (NLG), the Verbosity Manager, the ASR Output Simulator, the Patience Manager and the Natural Language Understanding module (NLU). The ASR Output Simulator communicates the N-Best corresponding to the current partial hypothesis to the Scheduler whose responses are conveyed to the NLU module.

3.1 Service task

The service used in our experiments is a personal agenda manager. The user can add events to the agenda, modify their attributes or delete them. To complicate a bit the task and justify the need for interactions a constraint has been introduced: all events must have separate time slots. If the user tries to overload a busy time slot, a warning is generated and the user is required to modify her request.

The simulated dialogue scenarios are defined by two event lists. The first one (*InitList*) is the list of events that already exist in the agenda before the dialogue and the second one (*ToAddList*) is the list of events, with priorities and alternative times, to add during the dialogue. The simulated user tries to make the maximum number of events with the highest priority values fit into the agenda. For example, if *InitList* contains the event $\{$**title:** *house cleaning*, **date:** *January* 6th, **slot:** *from 18 to 20*, **priority:** *3*, **alternative 1:** *January* 7th*, from 18 to 20*, **alternative 2:** *January* 9th*, from 10 to 12*$\}$ and *ToAddList* contains the event $\{$**title:** *birthday party*, **date:** *January* 6th, **slot:** *from 18 to 23*, **priority:** *2*$\}$ then the user simulator will first try to schedule his birthday party on January 6th from 18 to 23 but as a consequence, it will get a warning from the system because this slot is already booked for the house cleaning event. Therefore, the user simulator will reschedule the latter to January 7th from 18 to 20. If the house cleaning event was given with no alternative, it will be deleted as it has a lower priority compared to the birthday party.

3.2 Service implementation

The service is designed to process slot-filling requests in order to manage the user's personal agenda. The required slots are the *action type*, the event *description*, its *date* and its *time slot*. A keyword-based NLU module is configured to parse natural language requests.

A mixed initiative strategy is used for the interaction. First, the user takes the lead and utters a request in natural language. The system picks as many information slots as it can and if it still needs more information, it asks the user to complete her request slot by slot. In the following, a *dialogue task* refers to the portion of dialogue corresponding to a particular add, modify or remove action. In the best case, it consists of two dialogue turns only (request and confirmation) but if there is important noise, it might take several turns to complete (given that the user is patient enough not to hang up before the end of the dialogue task).

The following dialogue example is composed of two dialogue tasks. In the first one, the system manages to catch all the information needed to answer whereas in the second one, it missed the time slot and has to ask for it again. Moreover, the first dialogue task led to a conflict declaration whereas the second one could be accomplished:

- **System:** Hi! Welcome to your personal agenda service. How may I help you?
- **User:** Can you please add the event football game on November 9th from 18 to 20 if possible?
- **System:** Sorry. The time slot from 18 to 19 on November 9th is already booked for the event guitar course.
- **User:** Ok well, then move the event guitar course to November 10th from 18 to 19.
- **System:** Please specify a time slot.
- **User:** From 18 to 19.
- **System:** Great. So you want to move the event guitar course to November 10th from 18 to 19. Is that right?
- **User:** Yes.
- **System:** Good. Do you want to add another modification to your agenda?
- **User:** No.
- **System:** Bye.

3.3 ASR N-Best generation

Given *ToAddList* and the system responses, the Intent Manager is able to compute the user's immediate goal. Afterwards, the NLG transforms it into a sentence like *add the event birthday party on January* 6^{th} *from 18 to 23*. Moreover, if the simulated user takes the lead and starts asking for a specific slot, it is also able to give the information directly.

In (Ghigi et al., 2014), a corpus study shows that the user is likely to use off-domain expression, to add unnecessary information and to repeat the same piece of information several times in the same sentence (especially after a misunderstanding). To simulate this phenomenon, the user's request in natural language is given to a Verbosity Manager module that adds prefixes like *I would like to* and suffixes like *if possible*, that repeats the same information after a misunderstanding (with a given probability, e.g. 0.3) and that replaces the request with an off-domain sentence (with a given probability, e.g. 0.1).

To our knowledge, apart from the simulator described in (Selfridge et al., 2012), existent user simulators are turn-based and therefore, only the user intent is communicated at each turn (in a concept format) so there is no need to take care of the utterance formulations. This is not the case when incremental dialogue and turn-taking are the object of interest. In this case, the user's sentence is processed chunk by chunk. The update step is called a *micro-turn* and in this paper, the unit chosen is the word. Suppose that the current user utterance contains N words $w_1, w_2, ...w_N$ then at micro-turn t, the Verbosity Manager sends w_t to the ASR Output simulator. The latter stores an N-Best list from the previous micro-turn $\{(s_1^{(t-1)}, hyp_1^{(t-1)}), \ldots, (s_N^{(t-1)}, hyp_N^{(t-1)})\}$ that is updated according to w_t and WER (hyp_i is the i^{th} hypothesis in the N-Best and s_i is the corresponding confidence score). w_t can be replaced by a new word from a dictionary, deleted or a new word can be added to simulate the ASR noise, before it is added to the N-Best list.

The confidence score associated with the new word is computed as follows: if the word has not been modified, X is sampled from a Gaussian with mean 1 and variance 1 otherwise the mean is -1. We then compute the $sigmoid(X) = (1 + exp(-X))^{-1}$ as a word score (Figure 2 represents these two symmetric distributions). This

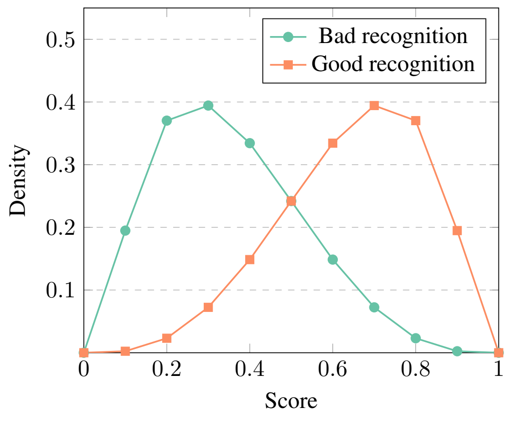

Figure 2: ASR score sampling distribution

is an adaptation of the simple ASR model introduced in (Pietquin and Beaufort, 2005). The score of the current partial utterance is the product of its words.

Another important aspect of incremental ASR is instability. A new ASR input does not necessarily translate into adding elements on top of the current output as it can change an important part if not the totality of it. For instance, in (Schlangen and Skantze, 2011), when the user says *forty*, it is first understood as *four* then *forty*. This is due to the fact that, given the language model, the new received chunk of information is more likely to complete a hypothesis that has a lower score in the N-Best list than the best hypothesis. In this work, as no language model is used, we use the NLU knowledge instead. If a new input leads to a new NLU key concept, then its score is boosted like in the following

$$s_i \leftarrow s_i + BF.(1 - s_i) \tag{1}$$

where the BF parameter (Boost Factor) is set to 0.2 in this work.

3.4 Time management and patience

In order to evaluate the time spent during the current dialogue, a speech rate of 200 words per minute is used (Yuan et al., 2006). Moreover, when the user hands the floor to the system, a silence of 2 seconds is added to this duration and 1 second the other way around. Finally, a Patience Manager module simulates the user patience: the maximum duration per dialogue task that the user can bear before hanging up. At each dialogue task, this value is computed as

$$d_{pat} = 2\mu_{pat}.sigmoid(X) \tag{2}$$

where μ_{pat} is the mean value (μ_{pat} = 180s).

3.5 Scheduler Module

A non-incremental dialogue system can be transformed into an incremental one by adding an extra module: the Scheduler (Khouzaimi et al., 2014). Its objective is to make turn-taking decisions (whether to take the floor or not). When the user speaks, its partial utterance grows over time. At each new change, it is sent to the Scheduler that immediately asks the service for a corresponding response and then rollbacks the system's context as long as it decides not to take the floor. If, on the other hand, it decides to commit to the last received partial utterance by taking the floor, then no rollback is performed and the dialogue context is effectively updated.

In this work, the Scheduler can perform two types of actions: WAIT and SPEAK, that is to say that it can wait for the next micro-turn without uttering anything or it can start retrieving the last response it got from the service.

Two versions of the Scheduler have been implemented: handcrafted rules were implemented in the first one whereas the second one embeds a reinforcement learning algorithm that learns to make turn-taking decisions by itself.

4 Turn-taking model

4.1 Turn-taking phenomena

Several turn-taking phenomena can be observed when analysing human conversations. A taxonomy of these phenomena is introduced in (Khouzaimi et al., 2015b), three of which are replicated here through the SPEAK action:

FAIL_RAW: The listener sometimes does not understand the speaker's message because of noise or unknown vocabulary. Therefore, she can barge-in and report the problem without waiting for the speaker to finish her sentence.

INCOHERENCE_INTERP: Unlike the previous phenomenon, in this case the listener fully understands the speaker's partial utterance. However, its content is considered problematic given the dialogue context and this can be reported immediately without waiting for the end of the utterance (system barge-in).

- Hi, I would like to book a room tonight and I...
- Sorry but there are no rooms available at the moment.

BARGE_IN_RESP: If the listener thinks she has all the information she needs to formulate a response, she can barge-in immediately which is frequent in human-human conversations.

4.2 Rule-based model

The three phenomena described above are replicated as handcrafted rules that have been implemented in the Scheduler:

FAIL_RAW: Depending on the last requested information by the system, it sets a threshold on the number of words. Whenever reached if the system still does not get any interesting information, it barges-in to warn the user about the problem:

1. Open question: this phenomenon is triggered if no action concept is detected (add, modify or delete) after 6 words (taking into account that the user can utter a prefix and leaving a margin because of the ASR instability).
2. Yes/no question: the threshold is set to 3.
3. Date question: it is set to 4.
4. Time slot question: it is set to 6.

INCOHERENCE_INTERP: An incoherence is detected in the user's utterance in the two following cases:

1. The user tries to fill a time slot that is already occupied.
2. The user tries to modify or delete a non-existing event.

Because of the ASR instability, as a security margin, the SPEAK decision will be taken two words after the incoherence is detected if it is maintained.

BARGE_IN_RESP: As soon as the service gives a full response to a partial utterance, the Scheduler considers that all the information needed has been given by the user. Like in the previous case, the decision is taken two words after.

4.3 Reinforcement learning

Despite late 20^{th} century initial proposition (Levin and Pieraccini, 1997), reinforcement learning as the machine learning framework in the field of spoken dialogue systems is still largely explored in the current days (Lemon and Pietquin, 2007; Laroche et al., 2010; Pinault and Lefèvre, 2011; Ferreira and Lefevre, 2013; Young et al., 2013). In non-incremental systems, at each dialogue turn, the system has to make a decision (action) hence moving to a new state. In this paper, as we study dialogue from a turn-taking point of view, the decision unit is the micro-turn.

4.3.1 Background

The turn-taking problem is here cast as a Markovian Decision Process (MDP) (Sutton and Barto, 1998), that is to say a quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the set of states where the system can be during a dialogue and $\mathcal{A}$ is the set of actions that can be performed at each time step. $\mathcal{T}$ is the transition model, in other words, the set of probabilities $\mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$ of getting to state s' at time $t + 1$ if the system was at state s at time t and performed action a. Such a decision makes the system get an immediate reward $r_t = R(s_t, a_t, s_{t+1})$ modeled by $\mathcal{R}$. The action to choose at each state is given by a policy π ($\pi(s_t) = a_t$) and the cumulative (discounted) reward is defined as $R_t = \sum_{t' \geq t} \gamma^{t'-t} r_{t'}$ (γ is called the discount factor). Finally, each couple (s, a) is associated with a value $Q^\pi(s, a) = \mathbb{E}[R_t|s_t = s, a_t = a]$ which is the expected cumulative reward for being at the state s, taking action a and following the policy π afterwards.

The goal of reinforcement learning is to find an optimal policy π^* such that, for every other policy π, and for every state-action couple (s, a), $Q^*(s, a) = Q^{\pi^*}(s, a) \geq Q^\pi(s, a)$.

4.3.2 State representation

The system state is characterised by the following features:

- **SYSTEM_REQ:** The current information that is asked for by the system. It can be a slot value, a confirmation or the system can ask an open question to make the user fill all the slots in one dialogue turn (6 alternatives).
- **LAST_INCR_RESP:** The Scheduler incrementally gets responses from the service. This feature corresponds to the last response obtained (11 alternatives).
- **NB_USER_WORDS:** The number of words added by the user after the last change in the value of LAST_INCR_RESP (after the last input that made the service catch a new piece of information and change its mind about the response to deliver).
- **NORMALISED_SCORE:** The ASR Output simulator estimates the score of a partial utterance as the product of its components. Therefore, the longer the sentence the worse the confidence score, even if all the components have a decent score. To neutralise this effect we normalise the score by taking its geometric mean given the number of words. Suppose there are n words in the current partial utterance and s its score, then NORMALISED_SCORE = $s^{\frac{1}{n}}$.
- **TIME:** The duration in seconds reached so far in the current task. This value is normalised so that it is around zero at the beginning of the task and around 1 for 6 minutes (maximum user patience).

In order to represent the function $Q(s, a)$, we maintain one linear model per action. There are 21 combinations between SYSTEM_REQ and LAST_INCR_RESP values that are the most likely to occur. They have been associated to the features δ_1 to δ_{21}. δ_i equals 1 when the i^{th} combination happens in the current micro-turn and 0 otherwise. Less frequent combinations have been removed from this initial model: first they make the model more complex with in all likelihood no significant improvement in the performance (making the learning process slower to converge and more data demanding) and second, the Fitted-Q algorithm involves the inversion of a feature covariance matrix which could be ill-conditioned with these rare combinations.

NB_USER_WORDS is represented by three Radial Basis Function (RBF) features (Sutton and Barto, 1998) ϕ_1^{nw}, ϕ_2^{nw} and ϕ_3^{nw}. Their means are set to 0, 5 and 10 and the corresponding standard deviations are 2, 3 and 3. The same representation with 2 features is used for NORMALISED_SCORE: ϕ_1^{ns} and ϕ_2^{ns} centered at 0.25 and 0.75 and with a standard deviation of 0.3 for both.

Finally, TIME is represented as a single feature $T = sigmoid((TIME - 180)/60)$ so that it is almost 0 for TIME=0 and almost 1 after 6 minutes. As this variable increases, the user is more and more likely to hangup, therefore the Q function is supposed to be monotonous with respect to that feature so it is taken directly in the model without the use of RBFs.

As a consequence, 28 parameters are involved for each action (56 in total). Let $\Theta(a)$ be the parameter vector ($\Theta(a) = [\theta_0, \theta_1, ..., \theta_{27}]^T$) corresponding to action a and $\Phi(s, a)$ the feature vector corresponding to state s and action a, therefore:

$$\begin{aligned} \Phi(s,a) &= [1, \delta_1, ..., \delta_{21}, \phi_1^{nw}, \phi_2^{nw}, \phi_3^{nw}, \\ &\quad \phi_1^{ns}, \phi_2^{ns}, T]^T \quad (3) \\ Q(s,a) &= \Theta(a)^T \Phi(s,a) \quad (4) \end{aligned}$$

4.3.3 Learning

RL learning of the turn-taking policy is operated with Fitted-Q, a Fitted Value Iteration algorithm (Lagoudakis and Parr, 2003; Chandramohan et al., 2010). Fitted-Q is a batch learning algorithm for reinforcement learning. Standard Q-learning (Watkins, 1989) has also been tested as an online algorithm but unsuccessfully, which is compliant with previous works (e.g. (Daubigney et al., 2012)).

The optimal Q-function Q^* is known to be the solution of the Bellman optimality equation (Sutton and Barto, 1998):

$$\begin{aligned} Q^*(s,a) = \mathbb{E}_{s'|s,a}[R(s,a,s') \\ +\gamma \max_{a' \in \mathcal{A}} Q(s', a')] \end{aligned} \quad (5)$$

Therefore, the Fitted-Q algorithm is fed with a set of N MDP transitions (s_j, a_j, r_j, s'_j) and aims to approximate the representation parameters vector of the Q-function by performing a linear regression at each iteration step. In our case, for each action and at the iteration i, the parameter vector is updated as follows (for commodity, $\phi(s_j, a_j)$ is noted ϕ_j):

$$\Theta^{(i)}(a) = \arg\min_{\Theta(a)} \sum_{j=1}^{N} (R_j^{(i-1)} - \Theta(a)^T \phi_j)^2 \quad (6)$$

$$R_j^{(i-1)} = r_j + \gamma \max_{a \in \mathcal{A}} (\Theta_{i-1}^T \phi(s'_j, a)) \quad (7)$$

This is a classical linear regression problem and we use the closed formula for each iteration:

$$\Theta^{(i)} = (\sum_{j=1}^{N} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{N} \phi_j R_j^{(i-1)} \quad (8)$$

The iteration stop criterion is

$$\sum_{a \in \mathcal{A}} ||\Theta^{(i)}(a) - \Theta^{(i-1)}(a)||_1 \leq \xi \quad (9)$$

In our experiments, the convergence threshold is set to $\xi = 0.01$.

5 Experiment

5.1 Experimental setup

Three dialogue scenario types involving diverse adding, modifying and deleting tasks were used for the experiments. For the training of the RL strategy, the simulated speech recognition WER was fixed at 0.15. We trained the system 50 times and each training session is made of 3000 episodes. The Fitted-Q algorithm was run every 500 episodes on the total batch from the beginning. During the first 500 episodes, a pure-exploration policy is used: it performs a WAIT action with a probability of 0.9 (hence a SPEAK action 10% of the times). An ϵ-greedy ($\epsilon = 0.1$) policy is then used until episode 2500. After that, a greedy policy is used (pure-exploitation).

Thus, 50 learning models are collected. As a linear model is used for the Q-function representation, we simply average the parameters to get an average model. The latter is then tested against the basic non-incremental strategy and our handcrafted baseline under different noise conditions by varying the WER parameter between 0 and 0.3 with a step of 0.03.

5.2 Results

The average learning curve is depicted in Figure 3. The reward levels corresponding to the non-incremental case and the handcrafted incremental strategy are indicated by the red and the blue lines. Each green triangle corresponds to the moving average reward over 100 episode of the RL strategy. The first 500 episodes are exclusively exploratory, therefore the system performance during that early stage of learning is pointless. Between episode 500 and episode 2500, we observe no improvement even though the policy is still partially exploring. This shows that the 500

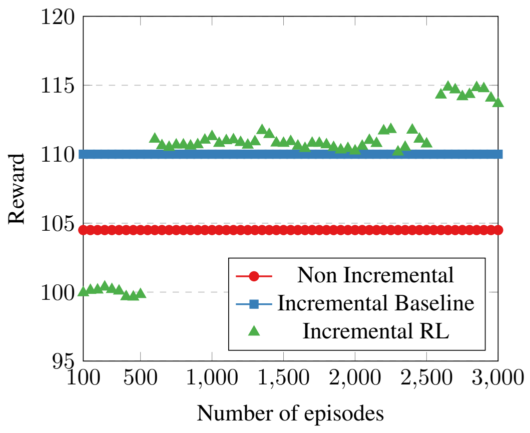

Figure 3: Learning curve (0-500: pure exploration, 500-2500: exploration/exploitation, 2500-3000: pure exploitation)

first episodes are enough to learn the optimal policy given our model. The 500 last episodes show that the learned strategy significantly outperforms the handcrafted baseline.

Incremental dialogue systems have the ability to report an error to the user in a more reactive way and to prevent it from speaking for a long time without being understood by the system. In noisy environments, these problems are even more likely to happen. Figures 4 and 5 show the effect of noise over dialogue duration and task completion. They represent the average performance over the 3 dialogue scenarios used in this experiment. Incremental dialogue, and the automatically learned strategy in particular, significantly increase the noise robustness. In the non-incremental case, the mean dialogue duration reaches 3 minutes and the task completion drops below 70%. Our learned strategy makes the dialogues finish 30 seconds earlier on average (17% gain) and the task completion is about 80%.

6 Conclusion and future work

In this paper, a simulated environment for a slot-filling task has been used to compare different expert and learned turn-taking strategies. A first one is non-incremental meaning that the user and the system cannot interrupt each other. As ASR noise increases, the dialogues tend to last longer leading to lower task completion. A second strategy is incremental, starting from three turn-taking phenomena present in the human-human interaction we translated them into a set of handcrafted rules for human-machine dialogue. This rule-based strategy is then shown to have better performance than the non-incremental case in terms of dialogue duration and task completion ratio when the noise is increasing. Eventually the Fitted-Q algorithm has been retained to automatically learn a third turn-taking strategy, still with the same objective (minimising the dialogue duration and maximising the task completion). This third strategy significantly improves noise robustness of the simulated dialogue system. We are now planning to evaluate this approach with real users and by taking subjective scores into account for learning optimal turn-taking strategies with respect to enlarged view of the system performance, such as comfort of use, friendliness etc.

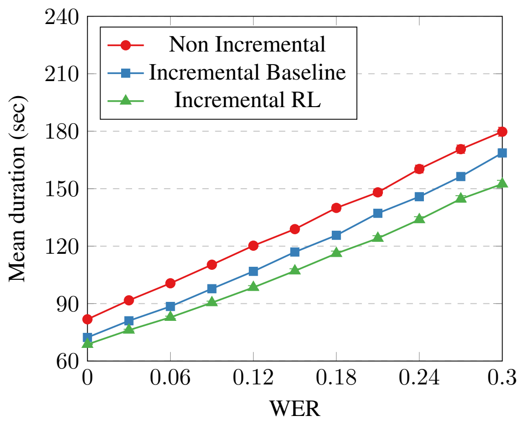

Figure 4: Simulated dialogue duration for different noise levels

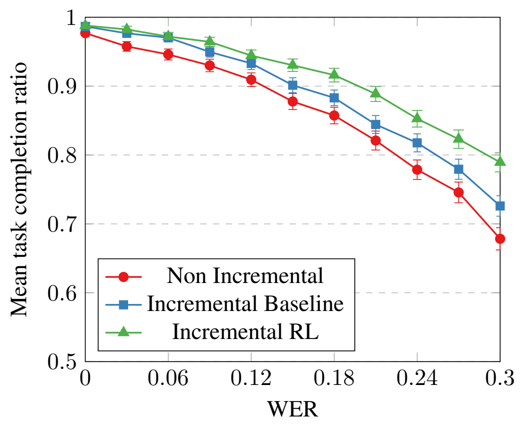

Figure 5: Simulated dialogue task completion for different noise levels

References

Senthilkumar Chandramohan, Matthieu Geist, and Olivier Pietquin. 2010. Optimizing spoken dialogue management with fitted value iteration. In *INTERSPEECH 11th Annual Conference of the International Speech*.

S. Chandramohan, M. Geist, F. Lefèvre, and O. Pietquin. 2011. User Simulation in Dialogue Systems using Inverse Reinforcement Learning. In *Interspeech*.

L. Daubigney, M. Geist, S. Chandramohan, and O. Pietquin. 2012. A comprehensive reinforcement learning framework for dialogue management optimization. *Selected Topics in Signal Processing, IEEE Journal of*.

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

David DeVault, Kenji Sagae, and David Traum. 2011. Incremental interpretation and prediction of utterance meaning for interactive dialogue. *Dialogue and Discourse*, 2:143–170.

W. Eckert, E. Levin, and R. Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*.

Emmanuel Ferreira and Fabrice Lefevre. 2013. Expert-based reward shaping and exploration scheme for boosting policy learning of dialogue management. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 108–113. IEEE.

Fabrizio Ghigi, Maxine Eskenazi, M Ines Torres, and Sungjin Lee. 2014. Incremental dialog processing in a task-oriented dialog. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Helen Hastie, Marie-Aude Aufaure, Panos Alexopoulos, Heriberto Cuayáhuitl, Nina Dethlefs, Milica Gasic, James Henderson, Oliver Lemon, Xingkun Liu, Peter Mika, Nesrine Ben Mustapha, Verena Rieser, Blaise Thomson, Pirros Tsiakoulis, and Yves Vanrompay. 2013. Demonstration of the parlance system: a data-driven incremental, spoken dialogue system for interactive search. In *Proceedings of the SIGDIAL 2013 Conference*.

Gudny Ragna Jonsdottir, Kristinn R. Thorisson, and Eric Nivel. 2008. Learning smooth, human-like turntaking in realtime dialogue. In *In Proceedings of Intelligent Virtual Agents (IVA 08*, pages 162–175. Springer.

Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. 2014. An easy method to make dialogue systems incremental. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. 2015a. Dialogue efficiency evaluation of turn-taking phenomena in a multi-layer incremental simulated environment. In *Proceedings of the HCI International 2015 Conference (accepted)*.

Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. 2015b. Turn-taking phenomena in incremental dialogue systems. In *Proceedings of the EMNLP 2015 Conference (submitted)*.

Dongho Kim, Catherine Breslin, Pirros Tsiakoulis, Milica Gasic, Matthew Henderson, and Steve Young. 2014. Inverse reinforcement learning for micro-turn management. In *INTERSPEECH Proceedings*.

Michail G. Lagoudakis and Ronald Parr. 2003. Least-squares policy iteration. *JOURNAL OF MACHINE LEARNING RESEARCH*.

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

Oliver Lemon and Olivier Pietquin. 2007. Machine learning for spoken dialogue systems. In *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *In EUROSPEECH 97*.

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

Olivier Pietquin and Richard Beaufort. 2005. Comparing asr modeling methods for spoken dialogue simulation and optimal strategy learning. In *INTERSPEECH*.

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

F. Pinault and F. Lefèvre. 2011. Unsupervised clustering of probability distributions of semantic graphs for pomdp based spoken dialogue systems with summary space. In *IJCAI 7th KRPDS Workshop*.

Antoine Raux and Maxine Eskenazi. 2008. Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In *SIGDIAL*.

Antoine Raux and Maxine Eskenazi. 2012. Optimizing the turn-taking behavior of task-oriented spoken dialog systems. *ACM Trans. Speech Lang. Process.*

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

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

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

Ethan O. Selfridge, Iker Arizmendi, Peter A. Heeman, and Jason D. Williams. 2012. Integrating incremental speech recognition and pomdp-based dialogue systems. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, July.

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

Christopher John Cornish Hellaby Watkins. 1989. *Learning from Delayed Rewards*. Ph.D. thesis, King's College.

S. Young, M. Gasic, B. Thomson, and J.D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*.

Jiahong Yuan, Mark Liberman, and Christopher Cieri. 2006. Towards an integrated understanding of speaking rate in conversation. In *INTERSPEECH Proceedings*.