Incremental Human-Machine Dialogue Simulation

Chapter · December 2017

DOI: 10.1007/978-981-10-2585-3_4

3 authors, including:

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

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

Incremental Human-machine Dialogue Simulation

Hatim Khouzaimi[1,2], Romain Laroche[1], and Fabrice Lefèvre[2]

[1] Orange Labs, Châtillon, France,
[2] CERI-LIA, Univ. Avignon, France,
hatim.khouzaimi@gmail.com, romain.laroche@orange.com,
fabrice.lefevre@univ-avignon.fr

Abstract. This article introduces a simulator for incremental human-machine dialogue in order to generate artificial dialogue datasets that can be used to train and test data-driven methods. We review the various simulator components in detail, including an unstable speech recognizer, and their differences with non-incremental approaches. Then, as an illustration of its capacities, an incremental strategy based on hand-crafted rules is implemented and compared to several non-incremental baselines. Their performances in terms of dialogue efficiency are presented under different noise conditions and prove that the simulator is able to handle several configurations which are representative of real usages.

Keywords: Incremental dialogue · Dialogue simulation

1 Introduction

Spoken Dialogue Systems (SDSs) still offer poor turn-taking capabilities as, in general, the listener must wait for the speaker to release the floor before reacting. Building on the recent ASR processing delay reductions [1], a new research field has emerged where SDSs offer the ability to be interrupted and to interrupt the user at any point, allowing a more natural floor management between speakers. For this purpose, the system should be able to incrementally process the user's request without waiting for the end of the user's turn [2–4], like in human-human dialogue [5]. Such systems are called *incremental dialogue systems* [6].

Designing optimal dialogue management strategies based on hand-crafted rules only is a difficult task. Therefore data-driven methods, mostly using reinforcement learning [7], are now largely applied [8, 9]. However, these techniques require a big amount of dialogue data which are very costly to collect. A common way to deal with this problem is to build a user simulator that emulates real users' behaviours [10, 11]. The motivation behind this approach is to validate that learning algorithms scale well and that they can learn in a context that is similar to reality, while avoiding the costs caused by real corpora gathering. Moreover, even though the learnt strategies are not guaranteed to perform well with real users, they can be used as a source policy for online learning, in a Transfer Learning approach [12].

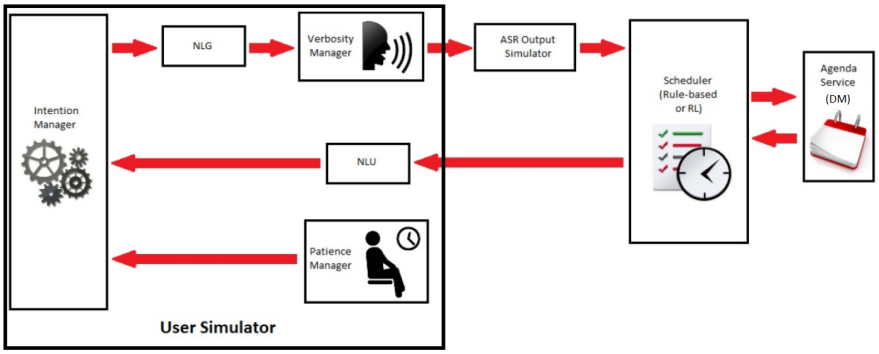

Fig. 1: Simulated environment architecture.

To the best of our knowledge, the only former proposition of an incremental simulated environment is in [13]. However it did not simulate the ASR instability phenomenon: as the ASR output is updated while the user is speaking, the best hypothesis is likely to be partially or totally revised. This is a critical problem in incremental dialogue processing as shown in [14, 15]. One must wait for last partial hypothesis to get stabilised before making a decision that is based on it. We show how to use information from the understanding module to emulate this mechanism.

The aim of this paper is to introduce an incremental user simulator in detail, so that it can be easily replicated by the reader. This simulator integrates the ASR instability phenomenon. A simple personal agenda management task is used here for illustration, however, the ideas developed here still hold on a wider scope. Moreover, some parameters are empirically set here, nevertheless, other values can be specified in order to test different configurations.

Section 2 presents the simulated environment and Section 3 describes a functioning illustration experiment. Finally, Section 4 concludes and provides an overview of future work.

2 Simulated environment

Our user simulator interacts with a dialogue system that is composed of a service and a Scheduler [16]. The first component handles high-level conceptual decisions whereas the second is in charge of floor management. An overview of the simulated environment architecture is given in Figure 1. Without loss of generality we will directly describe our simulator in light of the currently implemented service (personal agenda handling), allowing to give more practical cues on the implementations of the various components including parameter's true order of magnitude. Dialogues are in French, but they are translated here into English for the sake of clarity.

2.1 Service

Dialogue Manager (DM) The domain used for the simulation is an assistant that allows the user to add, modify or delete events in her personal agenda, as long as there is no overlap between them. In order to perform a task, the user must specify 4 slots: the action type (add, modify or delete), the event title (for example: *football game*), its date and its time window.

In its initial setting a mixed initiative strategy is used in order to gather theses pieces of information [17]. At first, the user is asked to formulate a request providing all the slot values and in the case there is still missing information (because of ASR noise or incomplete request), the system asks the user for the missing slots in turn.

Natural Language Understanding (NLU) The service parses the user simulator's utterances by iteratively applying a set of rules (in the order they appear in a rules file). As new concepts appear, some of them are regrouped in further iterations giving birth to other concepts. We show the details of parsing the sentence *I want to add the event birthday party on January* 6^{th} *from 9pm to 11pm* (at each step, the parsing output is provided followed by the applied rules):

1. **I want to ADD the TAG_EVENT birthday party on MONTH(January) NUMBER(6) from TIME(9,0) to TIME(11,0)**
   - add : [ADD]
   - event : [TAG_EVENT]
   - Regex(January|...|December) : MONTH($word)
   - Regex([0-9]+) : NUMBER($word)
   - Regex((([0-1]?[0-9])|(2[0-3]))h([0-5][0-9])?) : TIME($word)[3]
2. **I want to ADD the TAG_EVENT birthday party on DATE(6,1) WINDOW(TIME(21,0),TIME(23,0))**
   - Combine(NUMBER,MONTH) : DATE(NUMBER,MONTH)
   - Combine(from,TIME_1,to,TIME_2) : WINDOW(TIME_1,TIME_2)
3. **I want to ADD EVENT(birthday party, DATE(6,1), WINDOW(TIME(21,0),TIME(23,0)))**
   - Combine(TAG_EVENT,$x,on,DATE,WINDOW) : EVENT($x,DATE, WINDOW)
4. **I want ACTION(ADD, EVENT(birthday party, DATE(6,1), WINDOW(TIME(21,0),TIME(23,0))))**
   - Combine(ADD,EVENT) : ACTION(ADD,EVENT)

2.2 User simulator

Intent Manager The Intent Manager module computes the user simulator's next dialogue move at each turn. Its decision is based on an initial *dialogue scenario* defined by two lists of events: the events that are supposed to initially be present in the agenda before the beginning of the dialogue (*InitList*) and

[3] This rule is kept as an indication, it is the French way of telling time and it does not apply to English.

Table 1: A dialogue scenario example.

List	Title	Alternative	Date	Time window	Priority
InitList	house cleaning	0	January 6^{th}	18:00 to 20:00	3
		1	January 7^{th}	18:00 to 20:00	
		2	January 9^{th}	10:00 to 12:00	
ToAddList	birthday party	0	January 6^{th}	18:00 to 23:00	2

those the user simulator is supposed to add to the agenda during the dialogue (*ToAddList*). Moreover, each event is given a priority and potentially a list of alternative dates and time windows in case the main time window is not free. A dialogue scenario example is provided in Table 1 (we reduce each list to a single event to make it simpler to read, but it is not always the case).

The Intent Manager runs a recursive algorithm on a list of actions to perform: *actionList* (an action corresponds to either ADD, MODIFY, or DELETE). Initially, *actionList* is the list of the ADD actions corresponding to *ToAddList*. It starts by trying to perform the first action of the list (*actionList(0)*) and if it is successful, then this action is removed from *actionList* and the algorithm continues, until *actionList* is empty (stopping criterion). If the action could not be performed (because of a conflict with another event in the agenda for instance), the algorithm will try all the alternatives of this action (by successively adding the corresponding ADD actions in *actionList*) until one of them is successful. If it is still not the case, that means that each alternative alt_i is in conflict with a set S_i of events (most of the time, S_i contains only one element). Let m_i be the maximum priority of the events in S_i, then the algorithm focuses on the alternative indexed by $i_{min} = \arg\min m_i$[4]. If $alt_{i_{min}}$ has a lower priority than $m_{i_{min}}$ then it is discarded. Otherwise, the algorithm tries to recursively move the events in $S_{i_{min}}$ given their alternatives (the corresponding MODIFY actions are placed on top of *actionList*). The events in $S_{i_{min}}$ that do not have alternatives and the ones that could not be moved are deleted[5].

For instance, in the previous example, the Intent Manager will reschedule *house cleaning* to *January* 7^{th} *from 18:00 to 20:00* before adding the *birthday party* on *January* 6^{th} *from 18:00 to 23:00*.

Finally, during a dialogue, as long as the system declares misunderstandings, asks for specific slots that it did not catch or asks for a confirmation, the intent is constant. It only changes when the system fully understands that intent and confirms that fact. Therefore, the Intent Manager is able to provide an answer to an open question (where the user is supposed to provide all the slots at once)

[4] Here, the value of the priority variable designate its importance. Therefore, the higher the priority, the more important the task is.

[5] We do not claim that the Intent Manager algorithm solves the task in an optimal way. Moreover, there are some pathological examples that are not handled at all. Yet it complies with our objective to have a simple algorithm able to run realistic dialogues to study turn-taking mechanisms.

as well as intermediate dialogue acts (specific slot questions, misunderstandings and confirmations).

Natural Language Generation (NLG) At each dialogue turn, the user simulator knows which action has to be performed given the Intent Manager's output. The latter is then transformed into a simple straightforward request in natural language (NLG module), for instance: *add the event birthday party on January* 6^{th} *from 18:00 until 23:00* in the case of a response to an open question, *January* 6^{th} in the case where only the date is to be specified (same for the other slots) or *yes* after a yes/no question.

Then, the user's request is fed to an internal Verbosity Manager module. In [18], a corpus study with task-oriented dialogues showed that more than 10% of the users' requests were off-domain and that users often repeat the same information several times in a sentence especially after a misunderstanding. The Verbosity Manager outputs an off-domain sentence with a 0.1 probability, and when the system does not understand a request, it is repeated once with a 0.7 probability and twice in the same utterance with the remaining 0.3 probability (For instance, *January* 6^{th}*, I said, January* 6^{th}). Also, sentence prefixes such as *I would like to* and suffixes like *please* may be added.

Duration computation and Patience Manager To evaluate a dialogue strategy in the simulated environment, we estimate the mean dialogue duration as well as the task completion ratio. At the end of a dialogue, the total number of words is calculated and a speech rate of 200 words per minute is assumed to compute the dialogue duration [19]. A silence of one second is assumed at each regular system/user transition and a two seconds silence is assumed the other way round. In the case of interruptions and accurate end-point detection, no silence is taken into account.

The task completion rate is the ratio between the number of dialogues where the user did not hang up before a system's bye and the total number of dialogues. The user hangs up if the dialogue lasts for too long. For each new dialogue, the patience duration threshold is computed as $T_{max} = 2\mu_{pat} sigmoid(X)$ where X is sampled from a standard Gaussian distribution and μ_{pat} is the mean threshold ($\mu_{pat} = 180s$). The user hangs up as soon as the dialogue duration reaches that threshold. Real users' patience is complex and very difficult to model, hence, this approach is only a rough empirical approximation among other possibilities.

2.3 ASR output simulator

The output of the Verbosity Manager is fed to the ASR Output Simulator word by word. Each step is called a *micro-turn*. A scrambling function is used to simulate errors by replacing incoming words. The frequency of these replacements is following the Word Error Rate (WER) parameter which determines the noise level. Once a word w^* is elected for scrambling, the replacement acts as follows: with a 0.7 probability, a random word is taken from a dictionary (different from

w^*), with a 0.15 probability, a new word is added to w^* and with the remaining 0.15 probability, the word is deleted (like the other parameters specified in this paper, the values are empirical and they can be changed to model different configurations). At time t, the ASR output is an N-Best (best recognition hypotheses with their confidence scores) corresponding to all the words that have been pronounced so far $\{(s_1^{(t)}, hyp_1^{(t)}), (s_2^{(t)}, hyp_2^{(t)}), \ldots, (s_N^{(t)}, hyp_N^{(t)})\}$. At time t+1, a new word w_{t+1}^* pops up as an input and its N-Best[6] is calculated as follows (the best ASR hypothesis for the word w_t^* is called w_t):

1. Randomly determine whether w_{t+1}^* is among the N-Best with a specific probability. A parameter called INBF in $[0, 1]$ is used (In N-Best Factor, here set to 0.7): this probability is set to (1 - WER) + INBF.WER. If w_{t+1}^* is not among the N-Best, set the latter to a list of Scrambler samples and jump to step 4.
2. $w_{t+1} = w_{t+1}^*$ with a probability (1-WER). Otherwise, w_{t+1} is a Scrambler sample.
3. If the first hypothesis is the correct one, then the other $N - 1$ hypotheses are populated with Scrambler samples. Otherwise, the correct hypothesis position is taken randomly between 2 and N.
4. If the first hypothesis is the correct one, X is sampled from a standard Gaussian centered in 1. Otherwise, it is sampled from a standard Gaussian centered in -1. s_1 is then set to $sigmoid(X)$ (in order for it to be in $]0, 1[$). Following this method, correct scores are sampled from a distribution with a mean of 0.7, and 0.3 for incorrect scores. The standard deviation is 0.18 for both (it can be changed to simulate different confidence score model qualities, the closest to 0, the most discriminating the score confidence). The two distributions are drawn in Figure 2.
5. For each i between 2 and N, the scores are calculated in an iterative way: s_i is sampled uniformly between 0 and s_{i-1}.

In [13], the probability for outputting the correct word is taken as a basis for the confidence score. Here, a different point of view is adopted. It is an adaptation of the simple ASR introduced in [20] to the incremental case. Confidence estimation is a complex problem and it is still a research field [21, 22]. In addition, the confidence score is not always viewed as a probability, that is why the only assumption made here is that misrecognised words have a higher chance of having a score below 0.5 whereas well recognised words have scores above that threshold. This justifies the model presented on step 4 above. The score of the whole partial utterance is the product of the scores associated with all the $\tilde{w}_t$ that constitutes it (no language model is used for the sake of simplicity). Hence, at each micro-turn, the combinations that have the best scores form the N-Best corresponding to the current partial utterance.

[6] This N-Best corresponds to the last input word only. It is important to make the distinction between this N-Best and the one corresponding to the last partial utterance as a whole. In Figure 3, the block *New word N-Best* is a word N-Best whereas the other three blocks are partial utterances N-Best.

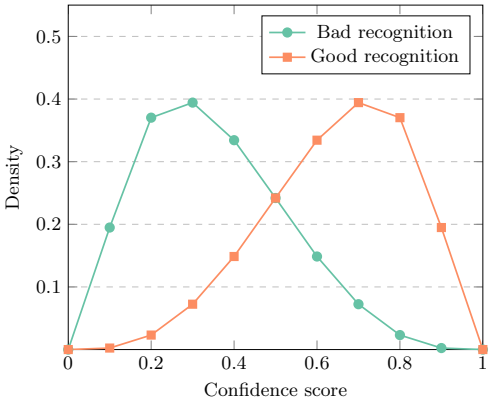

Fig. 2: ASR score sampling distribution

In real human-machine dialogue, a new increment in the ASR input can completely or partially change the best hypothesis (ASR instability [14, 15]). In order to simulate this instability, the scores of the hypotheses containing concepts that are correctly recognised by the Natural Language Understanding (NLU) module – thanks to the last ASR input – are boosted (a similar idea is used in the shift-reduce parser introduced in [23]). A parameter, called the *boost factor* (BF, here empirically set to 0.2) controls this mechanism: if a new NLU concept has just been recognised during the last micro-turn in the i^{th} hypothesis of the N-Best, then $s_i \leftarrow s_i + BF.(1 - s_i)$. An illustration of this mechanism is provided in Figure 3: at time t, the best hypothesis is *I want to add the event birthday party on mary* and at time $t + 1$ it becomes *I want to add the event birthday party on January* 6^{th}.

2.4 Scheduler

Functioning The Scheduler module principles have been introduced in [16] in order to transform a traditional dialogue system into an incremental one and at the same time, to offer an architecture where dialogue management in the traditional sense (dialogue act, conceptual decisions) is separated from turn-taking decisions. Here, the Scheduler is the interface between the user simulator (and the ASR Output Simulator) and the service. As mentioned previously the ASR Output Simulator sends growing partial results to the Scheduler. The latter requests the service immediately for a response and then picks an action among three options: to wait (WAIT), to convey the last service's response to the client (SPEAK) or to repeat the last word of the last stable partial request (REPEAT). It can be extended to repeating k words (REPEAT(k)). Because of the ASR instability, the Scheduler first removes some words at the end of the current hypothesis as a stability margin (SM) so that, if x_i is the i^{th} word of the current partial utterance, the decision is based only on $(x_1, ..., x_{n_t - SM})$ (called *the last stable partial request*) where n_t is the current number of words. So, the last SM words $(x_{n_t - SM+1}, ..., x_{n_t})$ will have time to get stable before being acted on. In

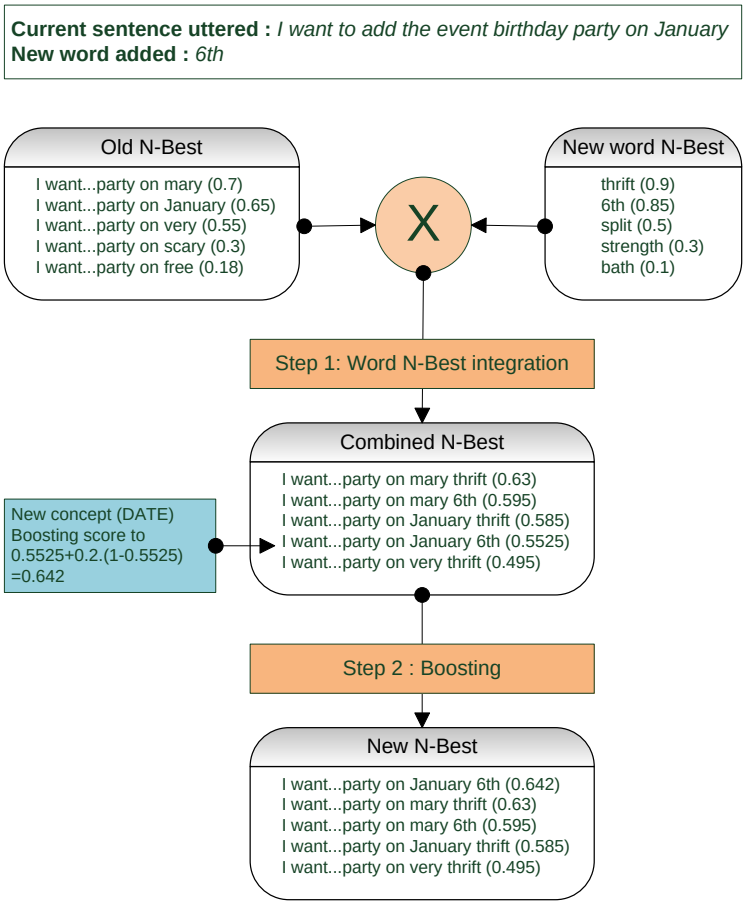

Fig. 3: Illustration of the N-Best incremental update

this work, we suppose that the speech rate is 200 words per minute [19]. So a word is pronounced in 0.3 seconds. In [14], a partial request that lasted for more than 0.6 seconds has more than 90% chance of staying unchanged. Hence, we set $SM = 2$.

3 Illustration

3.1 Turn-taking strategy example

Humans use a large set of turn-taking behaviours when talking to each other [24, 25]. In [26], we introduced a new taxonomy of these phenomena. The five turn-taking phenomena that are the most likely to improve the dialogue efficiency have been implemented in the Scheduler module of the simulated environment.

They are described (the labels from our taxonomy are used) in the following[7].
FAIL_RAW: The first phenomenon corresponds to the cases where the system interrupts the user because it could not understand the meaning of the partial utterance. Depending on the last system's dialogue act, a threshold relative to the number of words without detecting a key concept in the utterance has been set. In the case of an open question (where the systems waits for all the information needed in one request), if no action type (ADD, MODIFY or DELETE) has been detected after 6 words, a SPEAK is performed. The Scheduler waits for 3 words in the case of a yes/no question, for 4 words in the case of a date and for 6 words in the case of time windows (some concepts need more words to be detected and the user may use additional off-domain words).
INCOHERENCE_INTERP: Secondly, the system promptly reacts on partial requests that would lead to an error, not because they were not understood, but because they are in conflict with the current dialogue state. The Scheduler decides to speak as soon as the last stable partial request generates an overlap with another event in the agenda, or when the intent is to delete or modify a non existing event
FEEDBACK_RAW: When the system is not sure that it got the right message, it can repeat it. Here, when a new word is added to the partial utterance and the ratio between s_t and s_{t-1} is lower that 0.5, the Scheduler waits for SM words, and if the word is still in the partial utterance, a REPEAT action is performed.
BARGE_IN_RESP (System): Moreover, as soon as the user's sentence contains all the information needed by the system [27], the latter can immediately take the floor. That depends on the last system dialogue act as it determines which kind of NLU concept the system is waiting for. Once it is detected in the last stable partial request, the Scheduler performs a SPEAK.
BARGE_IN_RESP (User): The last phenomenon is the symmetric of the previous one [28, 29]. It is triggered directly by the user (no Scheduler decision is involved). The more familiar she is with the system, the earlier she will understand the system's intention when it has the floor, and the earlier she could interrupt. The moment where each dialogue act is interrupted is manually set.

3.2 Evaluation

In the first part of the experiment, the user simulator communicates directly with the service, with no Scheduler involved. Several non-incremental dialogue strategies have been simulated. In the System Initiative strategy (SysIni), the user is asked to fill the different slots (action type, description, date and time window) one by one. On the contrary, in the User Initiative strategy (UsrIni), she formulates one request that contains all the information needed to fill all the slots. In the Mixed Initiative strategy (MixIni), the user tries to fill all the slots at once, and if she fails, the system switches to SysIni to gather the missing information.

[7] Here, we only use the best hypothesis of the N-Best. However, the others are indirectly used through the boost mechanism.

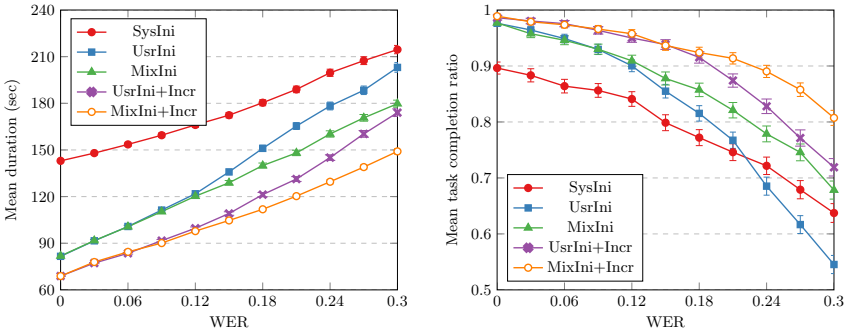

Fig. 4: Mean dialogue duration and task completion for generic strategies.

Three dialogue scenarios and different error rate levels were tested. For each strategy and each word error rate (WER), 1000 dialogues have been simulated for each scenario. Figure 4 represents the mean duration and the mean task completion, with the corresponding 95% confidence intervals, for the different strategies and for WER varying between 0 and 0.3.

SysIni is more tedious in low noise conditions. For WER above 0.24, its task completion becomes significantly higher than UsrIni even though its mean duration is still slightly higher (the duration distribution in the case of UsrIni is centered on short dialogues with a thick tail, whereas SysIni is more centered on average length dialogues). MixIni brings the best of each world as it behaves similarly to SysIni for low WERs and performs better than both SysIni and UsrIni in noisy environments.

In the second part of the experiment, the Scheduler is added between the user simulator and the service in order to simulate incremental dialogue. UsrIni+Incr (resp. MixIni+Incr) denotes the User Initiative (resp. Mixed Initiative) strategy with incremental behaviour (ability to interrupt the user through the four first phenomena described earlier and letting him interrupt the system through the last one). Incrementality is another way of making UsrIni more robust to noise, even better than MixIni. But, in noisy environment, the difference is slighter. Finally, combining the mixed initiative strategy and incrementality is the best strategy as MixIni+Incr achieves the same performance as UsrIni+Incr for low WERs but outperforms it for noisier dialogues.

4 Conclusion and future work

This paper describes an incremental dialogue simulator. The overview architecture as well as a detailed description of the different modules is provided, so that it can be easily replicated. Its functioning is illustrated through the implementation of an incremental strategy that replicates some human turn-taking behaviours in personal agenda management domain. Nevertheless, the framework introduced here is aimed to provide the reader with a general tool that can be adapted to different domains and different situations. It is particularly useful

for testing data-driven approaches as they require an important amount of training data which is costly when it comes to spoken dialogue systems. From the experiment results it can be observed that the proposed implementation truly offers a convenient way to simulate an improved floor management capacity and with parameters which will allow to adjust its global behavior to several targeted situations with different configurations.

The simulator has already been used to build a reinforcement learning model for optimising turn-taking decisions in order to improve dialogue efficiency [17]. In an on-going work, the turn-taking phenomena are tested with human users so as to validate that the simulator can effectively reflect real conditions. In future work, the data gathered from that experiment will be used to adjust the simulator's parameters in order to better replicate human behaviours.

5 Acknowledgment

This work is part of the FUI project VoiceHome.

References

1. Plátek, O., Jurčíček, F.: Free on-line speech recogniser based on kaldi asr toolkit producing word posterior lattices. In: Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL) (2014)
2. Allen, J., Ferguson, G., Stent, A.: An architecture for more realistic conversational systems. In: 6th international conference on Intelligent user interfaces (2001)
3. Dohsaka, K., Shimazu, A.: A system architecture for spoken utterance production in collaborative dialogue. In: Working Notes of IJCAI 1997 Workshop on Collaboration, Cooperation and Conflict in Dialogue Systems (1997)
4. Skantze, G., Schlangen, D.: Incremental dialogue processing in a micro-domain. In: Proceedings of the 12th Conference of the European Chapter of the ACL (EACL) (2009)
5. Tanenhaus, M.K., Spivey-Knowlton, M.J., Eberhard, K.M., Sedivy, J.C.: Integration of visual and linguistic information in spoken language comprehension. Science 268, 1632–1634 (1995)
6. Schlangen, D., Skantze, G.: A general, abstract model of incremental dialogue processing. Dialogue and Discourse 2, 83–111 (2011)
7. Sutton, R.S., Barto, A.G.: Reinforcement Learning, An Introduction. The MIT Press, Cambridge, Massachusetts, London, England (1998)
8. Levin, E., Pieraccini, R.: A stochastic model of computer-human interaction for learning dialogue strategies. In: The 5th biennial European Conference on Speech Communication and Technology (Eurospeech) (1997)
9. Lemon, O., Pietquin, O.: Data-Driven Methods for Adaptive Spoken Dialogue Systems. Springer Publishing Company, Incorporated (2012)
10. Eckert, W., Levin, E., Pieraccini, R.: User modeling for spoken dialogue system evaluation. In: Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on (1997)
11. Pietquin, O., Hastie, H.: A survey on metrics for the evaluation of user simulations. Knowledge Engineering Review (2013)
12. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research 10, 1633–1685 (2009)

13. Selfridge, E.O., Heeman, P.A.: A temporal simulator for developing turn-taking methods for spoken dialogue systems. In: Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (2012)
14. McGraw, I., Gruenstein, A.: Estimating word-stability during incremental speech recognition. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (Interspeech) (2012)
15. Selfridge, E.O., Arizmendi, I., Heeman, P.A., Williams, J.D.: Stability and accuracy in incremental speech recognition. In: Proceedings of the 12th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL) (2011)
16. Khouzaimi, H., Laroche, R., Lefèvre, F.: 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) (2014)
17. Khouzaimi, H., Laroche, R., Lefèvre, F.: Optimising turn-taking strategies with reinforcement learning. In: Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL) (2015)
18. Ghigi, F., Eskenazi, M., Torres, M.I., Lee, S.: Incremental dialog processing in a task-oriented dialog. In: Proceedings of the 15th Annual Conference of the International Speech Communication Association (Interspeech) (2014)
19. Yuan, J., Liberman, M., Cieri, C.: Towards an integrated understanding of speaking rate in conversation. In: 9th International Conference on Spoken Language Processing (Interspeech-ICLSP) (2006)
20. Pietquin, O., Beaufort, R.: Comparing asr modeling methods for spoken dialogue simulation and optimal strategy learning. In: 9th European Conference on Speech Communication and Technology (Eurospeech/Interspeech) (2005)
21. Jiang, H.: Confidence measures for speech recognition: A survey. Speech communication 45, 455–470 (2005)
22. Seigel, M.S., Woodland, P.C.: Combining information sources for confidence estimation with crf models. In: Proceedings of the 11th Annual Conference of the International Speech Communication Association (Interspeech) (2011)
23. Nakano, M., Miyazaki, N., Hirasawa, J.i., Dohsaka, K., Kawabata, T.: Understanding unsegmented user utterances in real-time spoken dialogue systems. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics (ACL) (1999)
24. Clark, H.H.: Using Language. Cambridge University Press (1996)
25. Sacks, H., Schegloff, E.A., Jefferson, G.: A simplest systematics for the organization of turn-taking for conversation. Language 50, 696–735 (1974)
26. Khouzaimi, H., Laroche, R., Lefèvre, F.: Turn-taking phenomena in incremental dialogue systems. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015)
27. DeVault, D., Sagae, K., Traum, D.: Incremental interpretation and prediction of utterance meaning for interactive dialogue. Dialogue and Discourse 2, 143–170 (2011)
28. El Asri, L., Lemonnier, R., Laroche, R., Pietquin, O., Khouzaimi, H.: NASTIA: Negotiating Appointment Setting Interface. In: 9th International Conference on Language Resources and Evaluation (LREC) (2014)
29. Selfridge, E.O., Arizmendi, I., Heeman, P.A., Williams, J.D.: Continuously predicting and processing barge-in during a live spoken dialogue task. In: Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL) (2013)