PhD THESIS

This dissertation is submitted for the degree of *Doctor of Philosophy*

SPECIALTY: Computer Science

Doctoral School 380 « Sciences et Agronomie »
Laboratoire d'Informatique (EA 931)

Reinforcement Learning Applied to Incremental Dialogue Systems

by
Hatim KHOUZAIMI

Publicly defended in front of a jury composed of:

Mme	Prénom NOM	Professeur, LABO, Ville	Rapporteur
M.	Prénom NOM	Maître de Conférences, LABO, Ville	Rapporteur
M.	Prénom NOM	Ingénieur, Entreprise, Ville	Examinateur
M.	Prénom NOM	Professeur, LABO, Ville	Examinateur
M.	Prénom NOM	Maître de Conférences, LABO, Ville	Directeur de thèse
M.	Prénom NOM	Professeur, LABO, Ville	Co-Directeur de thèse

Laboratoire d'Informatique d'Avignon

Contents

Introduction

Introduction here.

Part I

State of the art

Chapter 1

State of the art

Introduction

(This introduction is meant to be developed further and moved to the general thesis introduction section later. It will be replaced with a transition instead).

Defining the concepts constituting the thesis subject is the first step towards delimiting its frame. Here, simple hints and explanations are given in order to give the reader a first intuition of what *Reinforcement Learning Applied to Incremental Dialogue Systems* means. The rest of this chapter clarifies these ideas by providing more precise definitions and by grounding the manipulated notions in the current existing literature.

Originally, a **dialogue** designates a sequence of communication acts between two or more individuals through speech, either spoken or written (from Greek, *dia* means *through* and logos means *speech*). With the emergence of speech technologies, a research thread (to which this thesis belongs) developed machines that are able to substitute these individuals to a certain extent. They are made of a set of elements that are interacting with each other following precise rules, generally in the purpose of performing a specific task. As a consequence, they are referred to as **dialogue systems**.

Incrementalism is a method of work aimed to achieve a given task gradually, step by step. The adjective **incremental** designates any process that advances in that way. At each step, each new brick laid is called an increment. How is that related to dialogue systems? In a nutshell, an utterance is incrementally processed if the listener does not wait until its end before processing it, understanding it on the fly instead. As a result, these **incremental dialogue systems** can also utter words or sentences while the speaker is still holding the floor.

In computer science, **Learning** refers to the field of *Machine Learning* which is the science of building models that will drive algorithms to perform a certain task, and calibrating them automatically from data. **Reinforcement** is borrowed from the field of behavioral psychology. A behaviour can be strengthened in many ways, like being more frequently performed, for longer durations or after shorter delays for exam-

ple. This is generally due to a *positive stimulus* received by the agent under study, after adopting this behaviour. **Reinforcement Learning** is a mathematical framework with algorithms for solving problems through experience.

So, what is the point of applying reinforcement learning to incremental dialogue systems? What is the problem to solve in such systems? Traditional dialogue systems have only one kind of decision to take: what to say. Incremental dialogue systems, on the other hand, are free to speak whenever they want, which adds an extra dimension to the decisions it should take. In this thesis, decisions about the content of what the system should say are not studied, only timing decisions are focused on. Reinforcement Learning is therefore applied to investigate the following question: can an incremental dialogue system learn the proper timing for speaking by itself?

1.1 Human dialogue

1.1.1 Dialogue acts

If I say *This dog is big*, I utter a few sounds that can be cast as words. How are these words related to the real objects I refer to? How comes that a sequence of sounds can have effects on others? I can give orders to somebody and make them performs the actions I want as I can congratulate or insult someone and have an emotional impact on her. Also, how comes an utterance can also be judged as a complete nonsense or as a true or false assertion? These are a few questions raised in the philosophy of language.

In his book called *How To Do Things With Words* (Austin, 1962), J.L. Austin focuses on the concept of *speech act*, the title of another book (Searle, 1969) by John R. Searle, who completed this theory of language. Introducing these concepts is aimed towards bringing answers to the previous questions. Saying *My sister just arrived*, one performs a speech act that can be viewed from different points of view. Suppose that someone is listening as this sentence is being uttered and that person does not speak English, then obviously she only hears a sequence of noises which is the physical, low-level nature of the speech act. When considered from that perspective, the latter is referred to as a *phonetic act*. These sounds can then be put together to form a set of vocables and words which constitute a *phatic act*. When the whole meaning of the sentence is taken into account, the speech act is said to be a *rhetic act*. These three levels of analysis are grouped into a bigger concept, the *locutionary act*.

When one focuses not only on the raw meaning of a speech act but on the message that it contains, whether it is a warning or an encouragement for example, it is said to be an *illocutionary act* as beyond the rhetic aspect, an *illocutionary force* is considered. Nevertheless, in (Searle, 1968), Searle rejects this distinction between the rhetic and the illocutionary act made by Austin. According to him, the limit between the two is not justified hence rejecting the whole *locutionary act* concept. Instead, he suggests to adopt a modified classification. These considerations are, however, beyond the scope of this thesis as the philosophy of language will only be used as a tool to better analyse turn-

taking phenomena (see chapter 2). Therefore, only two concepts will be retained: the phonetic act where the speech act is viewed as a succession of sounds and the *illocutionary act* where the meaning is taken into account.

Moreover, saying something to somebody can have psychological effects on that person: congratulating someone or insulting him can be rewarding or hurting, a strong grounded speech can be convincing...etc... These are referred to as *perlocutionary acts*.

To build a dialogue systems, the traditional approach is to consider the user's and the system's speech acts as illocutionary (and sometimes perlocutionary) acts. When studying incremental dialogue systems, the *phonetic act* point of view comes into play. In this thesis, this distinction will be clarified.

In the field of psycholinguistics, Herbert H. Clark brings another view of dialogue acts in his book *Using Language* (Clark, 1996). The communication channel is split in two tracks, the *communication track* and the *meta-communication track*. The first one is used when the speaker adds new information about the topic of the information whereas the second one is used when she refers to the dialogue itself. For example, saying *OK, I see* or nodding her head are meta-communicative acts.

In order to guide the user throughout the dialogue, correct potential errors and to confirm some pieces of information, spoken dialogue systems use the meta-communication track. In this thesis, it is shown that incremental dialogue opens new possibilities to make an even more subtle use of this track.

1.1.2 Turn-taking in human dialogue

Turn-taking is a sociological phenomenon that has been generalised to many different situations: card games, road traffic regulation, CPU resource sharing...etc... The term *turn-taking* has been applied in that context for the first time both in (Yngve, 1970) and in Ervin Goffman's personal communications (June 5th 1970) independently. (Duncan, 1972) notices that *beyond considerations of etiquette, it is difficult to maintain adequate mutual comprehensibility when participants in a conversation are talking at the same time*. In (Sacks et al., 1974), Harvey Sacks describes the social organisation of turn-taking as an economy where *turns are valued, sought or avoided*, depending on the situation at hand. Then, in the rest of his paper, he focuses on the case of human conversation. Obviously, this is subject to many contextual and cultural variations but the objectif of the paper is to meet the challenge of extracting a set of rules that would ultimately describe the human turn-taking mechanisms in a general fashion.

For about six years, Harvey Sacks has been analysing conversation recordings and he came up with a few rules that caracterise human conversation and turn-taking. One of them is particularly interesting given the purpose of this thesis: *Transition (from one turn to a next) with no gap and no overlap are common. Together with transitions characterized by slight gap or slight overlap, they make up the vast majority of transitions*.

In fact, this principle has been brought to light even earlier in (Sullivan, 1947) where it has been noticed that the same turn-taking phenomena still hold over all the lan-

guages. (Schegloff, 1968) suggests that speaking *one party at a time* is one of the basic rules for conversations. In this thesis, however, it is shown that in task-oriented dialogue, it might be interesting to interrupt the speaker in some cases in order to increase dialogue efficiency.

During the following decades, a few other attempts to come up with models and classification of turn-taking phenomena in human dialogue have been made. In (Beattie, 1982), a political interview between Margaret Thatcher and Jim Callaghan has been analysed. As a result, the author introduced a classification of turn-taking phenomena where each category is caracterised by the answer to these three questions:

1. Is the attempted speaker switch successful?

2. Is there simultaneous speech?

3. Is the first speaker's utterance complete?

Optimising turn-taking means taking the floor at the right time. Humans are very good at detecting the cues for these timings. In artificial dialogue systems, different kinds of features can be used in order to detect these timings (Gravano and Hirschberg, 2011): prosodic features, lexical features, semantic features...etc... and a lot of work has already been done in this direction (Raux and Eskenazi, 2008; Jonsdottir et al., 2008; Meena et al., 2013).

Actually, humans even tend to start speaking before the end of their interlocutor's utterance, even interrupting him sometimes. A study led in (Strombergsson et al., 2013) reports statistics about times and overlaps in human conversation both directly and over the telephone. Given the type of question that is addressed to the listener, the latter tends to respond more or less quickly, often inferring the end of the question and starts uttering the response before its end. In (DeVault et al., 2011), the ability for the machine to guess the remaining part of a sentence while spoken is investigated.

In reality, human turn-taking involves even more complicated behaviours resulting in intertwined turns between the speakers. (Raux and Eskenazi, 2009) tries to provide a simple model with very few assumptions. It is a state machine where the following states are considered (initially depicted in (Jaffe and Feldstein, 1970)):

- Only the user speaks.

- Only the system speaks.

- No one speaks because the user stopped talking.

- No one speaks because the system stopped talking.

- Both the user and the system speak after a user barge-in.

- Both the system and the user speak after a system barge-in.

This is a very low-level model where only phonetic acts are considered (unlike the classification proposed in (Beattie, 1982) for example where the intent of the listener

when he tries to take the floor is taken into account). A similar approach is used in (Wlodarczak and Wagner, 2013), but the situations identified are different:

- **Within-speaker silence:** The speaker stops for while and then resumes his utterance.

- **Between-speaker silence:** The most intuitive case of turn taking. The speaker stops and after a moment of silence, the listener takes the floor.

- **Within-speaker overlap:** The listener either takes the floor or performs an intervention that is not meant to interrupt the speaker and the latter keeps the floor.

- **Between-speaker overlap:** The listener starts speaking before the end of the speaker's utterance, hence resulting in an overlap and a turn transition.

- **Solo vocalisation:** On person speaks in an unilateral way.

1.1.3   Incremental speech processing in human conversation

During a conversation, the listener does not wait for the speaker's utterance to end before trying to understand it. It is processed incrementally and on top of the intuition that people have related to this, a few studies provided convincing arguments supporting the idea. One of the most famous examples is an eye-tracking based study performed in (Tanenhaus et al., 1995). Subjects are given an image to look at through a head-mounted eye-tracking mechanism that records their eye-gaze at a millisecond scale. Then, ambiguous and unambiguous sentences were uttered, for example:

- **Ambiguous version:** Put the apple on the towel in the box.

- **Unambiguous version:** Put the apple that is on the towel in the box.

For this example, when the users are provided with an image of an apple on a towel, a towel with nothing on it and a box, their eye-gaze show different patterns depending on which version of the utterance they listen to. In the ambiguous case, they tend to start by looking at the apple, then to the towel with nothing on it, then to the apple again and finally at the box. In the unambiguous case, they directly look at the apple and then to the box. Most importantly, these movements happen as the sentence is uttered.

This is also pretty much related to the garden path sentence effect. Consider the following sentence (taken from the related Wikipedia article): *the government plans to raise taxes were defeated*. Most people feel the need to read it twice or even several times before understanding its meaning. When one starts reading *the government plans to raise taxes*, she automatically understands that taxes are planned to be raised by the government but then comes the disturbing end of the sentence: *were defeated*. The only solution is to parse *plans* as a noun and not as a verb. This is also an argument in favour of human incremental processing.

Finally, when humans are reading a text, they tend to skip a few words with no loss of the meaning. In (Ilkin and Sturt, 2011), the authors show that it is possible to predict a

few words while reading given the context and the sentence structure (eye-tracking has also been used here). For example, when the readers reach the sentence *The worker was criticised by his boss*, the word boss seems to be guessed ahead of time, again supporting the idea of incremental processing.

1.2 Spoken dialogue systems

A spoken dialogue system (SDS) is an automated application that interacts directly with a human being in natural language. Virtual assistant like Siri (Apple) or Cortana (Microsoft) are good examples of SDSs. They are task-oriented since their goal is to help the user achieve some task. There also exist a few SDSs that are only used for tutoring (Daubigney et al., 2013) or simply chatting and companionship (Sidner et al., 2013).

Historically, McCarthy and Hayes were the first to suggest the introduction of mental qualities (intentions, belief...) in a dialogue system (Mccarthy and Hayes, 1969; Mccarthy, 1979) in the 70's. This started a new research thread aimed at describing the internal state and the system's behaviour in a humanlike fashion (Newell, 1980; Bratman, 1987; Cohen and Levesque, 1990; Sadek, 1991; Konolige and Pollack, 1993). In Orange, a new dialogue system called ARTIMIS has been developed. It is based on a new theory of interaction (Sadek, 1991) based on the Belief-Desire-Intentions model introduced in (Bratman et al., 1988). During the following years it has been improved and enhanced with new capabilities: an action plan (Louis, 2002), a preference-based operator (Meyer, 2006) and uncertainty management capacities (Laroche et al., 2008).

However, this approach based on logic has been abandoned for two main reasons. Firstly, it does not guarantee the VUI-completeness (Pieraccini and Huerta, 2005; Paek and Pieraccini, 2008): it can lead to behaviours that has not been specified in advance, and secondly it requires an expertise in the logic field (which considerably reduces the span of potential dialogue system designers). As a result, like all the major actors in the field (Nuance, AT&T, SpeechCycle...etc...), Orange turned to a new solution where the dialogue system is viewed as an automaton. This solution is called Disserto which has proved to be simple, robust enough to allow the development of widely deployed industrial dialogue systems and flexible enough to allow state of the art research. All the services developed in this thesis have been generated using Disserto.

1.2.1 Towards a human-like interaction

Traditional computer interfaces are presented in the form of controls in an application (text boxes, buttons) or web pages. They are heavily used and they have been proven to be efficient enough, providing an accurate way of human-machine interaction. So, why building spoken dialogue systems? What are the advantages of the vocal modality?

An obvious motivation for building spoken dialogue systems is the quest for human-likeness. In (Edlund et al., 2008), an interesting analysis of the way humans perceive dialogue systems they are interacting with is performed. These systems are complex and

humans keep a biased representation of them during the interaction (called metaphors in the article). Two of them are the most common:

- *Interface metaphor:* The system is viewed as an interface just like non dialogue-based systems. Users adopting this representation of the system tend to use vocal commands instead of complete sentences. Moreover, they tend to keep silent when the system tries to behave like a human, by saying *Hi!* for example.

- *Human metaphor:* In this case, the users tend to view the system as a real human, therefore adopting a more natural way of communication. These users generally have higher expectations of the system's ability to answer their requests as well as its ability to perform human-like turn-taking behaviours.

Nevertheless, it is also legitimate to ask the question whether human-likeness should be a goal in the conception of dialogue systems or not. In (Edlund et al., 2008), four kinds of objections to pursuing that goal are discussed. The first one is the feasibility: is there any hope that someday, systems that behave exactly like humans could be built. This has raised huge debates during the last decades. The second question is utility: do people really need systems that imitate humans? Apart from the fact that it would help us to better understand the way humans communicate, it is easy to justify in the case of some applications like companionship and entertainment. It is less obvious when it comes to task-oriented dialogue. The third point is related to the concept of *uncanny valley* (Mori, 1970): as machine's human-likeness increases, they reach a point where they start becoming disturbing for the user. Nevertheless, by bringing even more improvements, this valley will be crossed hopefully and human-like solutions that no longer have such problems will be built. Finally, the last brought up question is the one of symmetry. If machines have a similar behaviour to humans, then users are likely to push the human metaphor to the limit by unconsciously thinking that machines really understand what they are saying, thus expecting more complex reactions like the ones due to emotions for example.

The multiplication of services and support platforms in modern society engenders huge costs that dialogue systems can help to reduce. By analysing client paths while demanding assistance from an expert, it is possible to identify commonly asked questions and recurrent patterns. By gathering such information, it is also possible to design dialogue systems that can interact with users and respond to their requests without any human intervention (of course, in the case the interaction fails, the client is redirected to a normal service platform). This can dramatically reduce service and support costs and also improve the quality of service as it is accessible any time, with no interruptions (unlike real platforms that are more often open at working time only). Moreover, when a client calls, the answer is immediate and her call is no longer queued causing waiting time (which she pays for) with sometimes no answer at all.

With the development of the Internet of Things (IoT) during the last few years, new human-machine interactions can be imagined. For instance, the concept of *Smart Home* is currently making its enter in the market of Artificial Intelligence through the contributions of several companies and start-ups: Amazon Echo... In this context, the advantage of speech communication is clearly relevant as it is hands and eyes-free. The

user can command her house from any room with no extra device needed. In some situations, her hands are already occupied by some other task. For example, she can be cooking (Laroche et al., 2013) while asking *What should I put next in my salad?* or *Can you add milk to my shopping list please?*.

Finally, as talking agents and talking robots can also be designed for entertainment and companionship (Sidner et al., 2013) and the vocal modality is the most natural way of interaction, it is very useful in this area.

1.2.2 Spoken dialogue systems architecture

The classic architecture of an SDS is made of five main modules (Figure 1.2):

1. Automatic Speech Recognition (ASR): transforms the user's audio speech signal into text.

2. Natural Language Understanding (NLU): outputs a conceptual representation of the user's utterance in text format.

3. Dialogue Manager (DM): given the concepts extracted from the user's request, a response (in a conceptual format too) is computed.

4. Natural Language Generation (NLG): transforms the concepts computed by the DM into text.

5. Text-To-Speech (TTS): reads the text outputted by the NLG by using a synthetic voice.

Automatic Speech Recognition

Speech recognition technology is an old problem with long history. During the 1950s, a group of researchers from Bell Labs developed the first technology that is able to recognise digits from speech (in fact, speech perception has been under study since the early 1930s). Then, during the second half of the last century, new advances have made it possible to build ASR solutions with larger vocabulary and with no dependence on the user. In the 1960s, Hidden Markov Models (HMMs) were proved to be useful for speech recognition (Gales and Young, 2007) and they were the most popular technique two decades later. Commercial products using ASR technology had to wait until the 1990s to be finally released in the marked as they reached an interesting vocabulary scope (even though their accuracy and their delay were far behind the technology available today). Performances kept improving slowly and gradually until 2009, when Deep Learning algorithms were tested (Mohamed et al., 2009; Deng et al., 2013) introducing huge improvement; the Word Error Rate (WER) decreased by 30%. During the last six years, research continued in that direction giving birth to accurate and reactive speech recognition solutions (Google, Nuance, Sphinx, Kaldi...). These solutions also provide results incrementally in a continuous fashion. Therefore, ASR is less and less considered as a bottleneck in the development of spoken dialogue systems, and as

Figure 1.1: *The dialogue chain*

```
┌──────────────────────────────────────────────────────────────┐
│                            5-Best                              │
│                                                                │
│   I would like to book a flight from New-York to Los Angeles (0.965) │
│   I could like to book a flight for New-York to Los Angeles (0.931)  │
│   Cold like a book fly New-York to Los Angeles (0.722)         │
│   Cold like look flight pork to Los Angeles (0.570)            │
│   Cold bike look fly New-York to Los Angeles (0.441)           │
└──────────────────────────────────────────────────────────────┘
```

Figure 1.2: *A 5-Best example corresponding to the sentence "I would like to book a flight from New-York to Los Angeles".*

it will be shown later, the delays they offer make it possible to design reactive incremental dialogue systems. Commercial off-the-shelf ASR solutions like Google ASR or Nuance products are able to recognise almost every word in many languages, including named entities. Finally, the ASR output is not only the text that the recognition algorithm figures out to be the best match for the input audio signal, but a list of the N most likely hypotheses and the corresponding confidence scores: it is called the N-Best. For instance, an 5-Best example is represented in Fig. 1.2.

Natural Language Understanding

NLU is a sub-field of Natural Language Processing (NLP) whose scope is wider than the spoken dialogue field. Since the 1950s, researchers have been trying to develop several models and ontologies in order to automatically process natural language with several applications in sight: topic recognition, sentiment analysis, news filtering and analysis, natural speech processing...etc...The ambition manifested during 1950s and the early 1960s quickly had to face reality as the expected objectives were far from being reached. As a consequence, research in this area was significantly slowed down between the 1960s and the 1980s. During the last decade, NLP research has found a second wind thanks to new Machine Learning techniques, bringing them at the heart of lucrative businesses like recommendation and digital marketing. NLU refers to the set of techniques in order to make the machine understand the underlying structure of a text in natural language. To do so, a lexicon as well as an ontology (concepts and the links between them in a specific domain) should be built. Therefore, earlier NLU solutions are based on a set of handcrafted parsing rules, however, new statistical-based models (Macherey, 2009) have been proven to be more robust and easy to generalise over domains. Also Deep Learning has also been applied to NLP showing interesting performances that NLU modules can benefit from (Bengio et al., 2003; Collobert et al., 2011).

Dialogue Management

As far as Dialogue Management is concerned, two decades ago, for the first time, dialogue has been modeled as Markov Decision Processes (MDPs) problem (see Sect. 1.4), hence being solved using reinforcement learning (Levin and Pieraccini, 1997). The dialogue state contains all the information needed to determine what is the best action to perform as well as the quality of that state (roughly speaking, to what extent it is desirable for the system to be at that state, in order to maintain the Markov property explained later). The possible actions in each state are the dialogue acts that the system can make while being at that state. In 2007, in order to represent the uncertainty over the user's intent (due to ASR or NLU imperfections), dialogues have been cast as Partially Observable Markov Decision Processes (POMDPs) (Williams and Young, 2007). This gave raise to the notion of *belief tracking* which objective is to keep a distribution over the possible user intents. Finally, existing dialogue systems are able to interact with the user in the domain they are built for only, however, during the last few years, researchers have been pushing the boundaries of domain extension methods (Gasic et al., 2013) and open-domain systems (Pakucs, 2003; Ekeinhor-Komi et al., 2014; Wang et al., 2014).

Natural Language Generation

The NLG task is the inverse of the NLU one: it translates dialogue acts' concepts into sentences. It started being used in the 1990s for purposes like financial news summary. A few start-ups and big companies also provide automatic text generation solutions that are used to quickly produce reports or official letters. The main challenge for such systems is to be able to generate a variety of different words, expressions and sentences in order for them not to be repetitive and for the result to be as realistic as possible. This is even more crucial when it comes to dialogue systems as they are supposed to simulate real conversations with the user, which is a highly variable. Nevertheless, in practice, template-based generation methods are the most widely used.

Speech Synthesis

During the 1930s, Bell Labs were not only interested in ASR but they also developed a new approach for the reciprocal task: the TTS (also known as *speech synthesis*). The human speech is broken into small acoustic components that are sequentially pronounced by the system. They built the first machine demonstrating this mechanism: the Voder (Dudley, 1939). As far as this task is concerned, the challenge for the system is to sound as human-like as possible, in terms of phoneme transitions, speech rate and prosody. Two research threads tackle this problem in two different manners (Tabet and Boughazi, 2011): the first one uses corpus-based methods where the resulting audio is a combination of pre-recorded phonemes extracted from a corpus and the second one uses HMMs to generate a completely synthetic voice. Even though substantial advances have been

accomplished since the Voder, it is still easy to distinguish between a synthesised and a real human voice.

1.2.3   Spoken dialogue systems evaluation

When building dialogue systems and improve them, it is necessary to determine metrics in order to measure their evolution. What makes a dialogue system better than another one? What metrics should be taken into account while evaluating dialogue systems? What are the most important characteristics of a dialogue system that should be improved?

In the PARADISE framework (Walker et al., 1997), a distinction is made between two kinds of metrics that are usually used for evaluating dialogue systems: objective and subjective metrics. The first category contains all the Key Performance Indicators (KPIs) that can be measured by an algorithm by accessing the dialogue logs only, whereas the second one is made of the user's appreciations of the dialogue quality or specific characteristics like human-likeness or to what extent the user enjoyed the dialogue experience.

Objective metrics that are commonly used are the dialogues' mean duration and the task completion ratio. Generally speaking, these two metrics are correlated as the user gets impatient when the dialogue lasts for too long (the user can also get impatient for other reasons, like the repetition of the same system's dialogue act several times). Moreover, the user's speech rate and the way they communicate introduces some variability when using these metrics. Finally, it is legitimate to ask ourselves: are shorter dialogues the real desired objective? First, this depends on the type of dialogue system at hand. If it is designed for entertainment or companionship, then there is no need for seeking faster dialogue strategies. However, in the case of task-oriented dialogue, looking for shorter dialogues makes sense as a measure of efficiency. In these situations and especially for daily tasks, an efficiency threshold has to be necessarily reached in order for people to use the dialogue system at hand.

Subjective metrics are generally gathered using a survey at the end of each dialogue or by experts rate dialogues afterwards(El Asri et al., 2014a). Several metrics can be collected this way: the global quality of the dialogue, naturalness/human-likeness, reactivity...etc...However, this also raises the problem of variability between users. Most often, they are asked to evaluate the system on a Likert scale (1 to 5 or 1 to 10), but a user answering 4 could be equivalent to another user answering 3 or 5. Therefore, the absolute evaluation is less significant than the relative one given a specific user.

(Complete and add more citations, see Layla's thesis).

1.2.4 Spoken Dialogue Systems at Orange

1.3 Incremental dialogue systems

1.3.1 Principles

Currently deployed dialogue systems have a simple and rigid way of managing turn-taking. The interaction mode they offer is similar to a walkie-talkie conversation as the system waits for the user to finish her utterance before taking the floor and vice-versa (even though some systems allow the user to interrupt them). Such systems will be referred to as *traditional dialogue systems* in this thesis.

The first idea of incremental systems goes back to incremental compilers (Lock, 1965). An incremental compiler processes each new instruction independently from the previous ones. Therefore, a local modification of the code does not affect the whole result of the compilation. The idea of processing natural language in an incremental way is first introduced in (Wirén, 1992) according to (Kilger and Finkler, 1995). Instead of feeding modules with complete utterances, the input is pushed chunk by chunk (500ms of audio signal, one word of a sentence...etc...) making the output change several times before the end of the user's utterance. Nevertheless, in his book *Speaking: From Intention to Articulation* (Levelt, 1989), Levelt analysed the mechanisms underlying the way humans formulate their ideas in natural language and already reported that the processes involved are incremental. The approach is closer to computational linguistics than psycholinguistics. The second part of the dialogue chain (DM, NLG and TTS) is analysed using a different terminology: the *Conceptualizer*, the *Formulator* and the *Articulator*.

As discussed in Sect. 1.1.3, in human to human conversation, the listener does not wait for the speaker to finish his sentence before processing it; it processes it as it is spoken. As a consequence, human beings perform a large panel of turn-taking behaviours while speaking, like backchanneling (*aha*, *yeah*...) or barging-in.

To replicate these behaviours, a new generation of SDSs has been the focus of research for the last few years. An SDS is said to be incremental when it is able to process the user's speech on the fly. The input signal is divided into small chunks and the growing sentence is reprocessed at each new chunk (Schlangen and Skantze, 2011). Table 1.1 gives an example illustrating the functioning of an incremental NLU module (in a hotel room booking service). For the sake of simplicity, processing delays are neglected.

1.3.2 Advantages of incremental processing

Before discussing the different reasons why incremental processing should be preferred to rigid turn-taking, it is important to note that a few studies with real users have shown that incremental dialogue systems offer a better user experience. For instance, in (Aist et al., 2007), an ordinal regression has been performed between the user satisfaction and

Time step	NLU input	NLU output
1	I	empty
2	I won't	empty
3	I would	empty
4	I would like to	empty
5	I would like to cook a	empty
6	I would like to book a room	**action:** BOOK
7	I would like to book a room on May	**action:** BOOK
8	I would like to book a room on May 7th	**action:** BOOK **date:** 05-07
9	I would like to book a room on May 17th	**action:** BOOK **date:** 05-17
10	I would like to book a room on May 17th and I will	**action:** BOOK **date:** 05-17
11	I would like to book a room on May 17th and I will be driving	**action:** BOOK **date:** 05-17 **parking:** YES

Table 1.1: *Example of incremental NLU processing*

several features with a flag for incremental processing among them. A significant correlation between incremental processing and the global user satisfaction has been found. Other studies also confirm the advantage of incremental speech processing (Skantze and Schlangen, 2009; El Asri et al., 2014b; Zhao et al., 2015).

As discussed in Section 1.2.1, human-likeness is a legitimate goal for dialogue systems (at least worth trying). When talking to each other, humans perform a rich set of turn-taking phenomena (see Section 1.1.2) and in spite of the fact that they do not talk in a rigid walkie-talkie manner, they manage to avoid desynchronisations and to keep a conversation that is going forward. Replicating these behaviours from the machine's point of view can therefore be interesting. It might be expected that the user feels more at ease while using a more human turn-taking mode hence pushing the human metaphor even further.

The other aspect that is interesting about incremental dialogue is reactivity. As the system processes the user's request before its ends, it is possible to design accurate end-point detection in order to detect the end of this request as soon as possible (Raux and Eskenazi, 2008). Moreover, incremental dialogue systems can interrupt the user to report a problem, like in the following example:

USER: I would like to book a room for tomorrow with ...

SYSTEM: Sorry, we are full tomorrow.

This can help the user get to her goal faster but one should be very careful about the way it is implemented as there is a risk that user interruption actually harms the user experience (even though the intent is to go faster, see Section 1.3.3). Nevertheless,

a corpus study led in (Ghigi et al., 2014) showed that when users are interrupted, they tend to adopt a more sober way of expression, hence directly increasing the dialogue efficiency but also indirectly as the risk of misleading off-domain words and expressions is reduced (Zhao et al., 2015).

Early barge-in both from the user and the system's side is also a way to limit desynchronisations. An example of a desynchronised dialogue could be (inspired by the NUMBERS domain described in (Skantze and Schlangen, 2009)):

USER: 01 45 38 37 89

SYSTEM: 01 45 28 37 89

USER: No, not 28 but 38

SYSTEM: Sorry, you mean 28 38

USER: What?

SYSTEM: 28 38 1 *(The system understood "One" instead of "What?")*

In this example, the user could have reported the mistake earlier if he could barge-in:

USER: 01 45 38 37 89

SYSTEM: 01 45 28...

USER: No, 38

SYSTEM: Ok. 01 45 38

USER: 31 89

SYSTEM: 31 89

Another interesting aspect about incremental dialogue systems is that they can leverage multimodality (Fink et al., 1998). In fact, there are two aspects of multimodality and they can both benefit from incremental processing:

- **Input multimodality:** Most researchers in the community refer to this aspect when talking about mutimodal systems. A system input can be multimodal in the sense that it can handle speech input, but also text, gesture or eye-gaze for example. The main challenge faced by this kind of setup is the problem of mixing these inputs in order to infer the correct user intent. In the case of multimodal systems, the world is considered as a flow of information coming from multiple sources of information (Chao and Thomaz, 2012; Rosenthal et al., 2013). The different modalities are not necessarily sampled with the same rate nor support the same delays, therefore, it is important to find convenient ways to synchronise them.

- **Output multimodality:** The machine can also use different channels of information while communicating (Matthias, 2009). For example, the speech modality can be used at the same time as a moving avatar with facial expressions (the Furhat

for example (Skantze and Johansson, 2015)). It can also be coupled with the input multimodality paradigm to create highly interactive interfaces like (Johnston et al., 2014).

1.3.3 New challenges raised by incremental dialogue processing

The first problem to consider when talking about incremental spoken dialogue systems is the question of ASR *latency*, which is the time needed by the recognition algorithm to provide the text output corresponding to an audio signal. As discussed earlier, the ASR accuracy has been a bottleneck in the development of spoken dialogue systems for many years but thanks to recent advances in this field, it is no longer the case. Similarly, incremental dialogue systems require quick responses from the ASR but speech recognition modules have been too slow for many years which was a limiting factor in the development of incremental dialogue processing. But, in the last few years, ASR technology has become reactive enough (Breslin et al., 2013; Plátek and Jurčíček, 2014). Still, it is important to be aware that there is a tradeoff between the accuracy, the vocabulary size and the latency. Kaldi, which is an ASR solution designed by researchers (and which is mostly used by them), makes it possible to design one's own acoustic and language model as well as setting one's own parameters in order to control this tradeoff. Off-the-shelf solutions like Google ASR do not give the user such possibilities (however, accuracy and delays in open domain are well balanced for most applications).

If the successive partial results of an incremental ASR module are observed during a user's utterance, it is likely that the progression they follow is not monotonous. In other words, a partial result is not guaranteed to be a prefix of results to come. The following example showing successive ASR results illustrates this phenomenon:

1. Euh

2. I

3. Euh good

4. iPod

5. I would

6. I good bike

7. I would like

This phenomenon is called ASR *instability* (or stability depending on the sources) (Selfridge et al., 2011). This factor is also related to the tradeoff between latency and accuracy as preferring fast ASR over accurate ones can lead to very unstable results (the system is not given enough time to compute accurate results most of the time, thus ending up delivering wrong partial results frequently), and vice-versa.

This leads to one of the main challenges raised by incremental processing: the ability to *revise* the current hypothesis. All the modules in the dialogue chain are impacted by

this problem. As an illustration, suppose that the user interacts with an incremental personal assistant on her phone and makes the following request *Please call the number 01 45 80 12 35*. The last digit is first understood as being 30 and then 35, therefore, if the system is too reactive, there is a risk that it starts calling the wrong number and maybe start uttering the sentence *Ok, calling 01 45 80 12 30*. Afterwards, the system understands 35 instead of 30 hence needing a correction mechanism in order to stop the TTS, to cancel the call, to perform a new one and to provide a new answer to the user. Nevertheless, even though the system at hand is equipped with such a mechanism, using it very often is not an optimal way of managing incremental input as it causes extra delay as well as non-natural behaviour (stopping the TTS and starting again with another utterance). This introduces a similar tradeoff to the one discussed for the ASR module but from the DM perspective: if decisions are taken too quickly, it is likely that some of them are wrong hence activating the correction mechanism. On the other hand, if the DM is slow to take action, then it lacks reactivity and there is no point for it to be incremental. As a consequence, it is important to determine the right moment to commit to the current partial utterance and to take action based on it (Raux and Eskenazi, 2008; Lu et al., 2011).

Incremental NLG also raises new problems which are illustrated in (Baumann and Schlangen, 2013). In this paper, a system has to describe a car's trajectory in a virtual world. When the latter approaches an intersection where it has to turn right or left (no road straight ahead), then the system utters something like *The car drives along Main Street and then turns...euh...and then turns right*. In this example, the system is sure that the car is going to turn which makes it possible for it to commit to the first part of the sentence with no risk. However, this is not always the case as a new chunk of information from the user can change the whole system's response. In this thesis, the NLG is not incremental as the DM's response is considered to be computed instantly at each new micro-turn (event though it is not necessarily stable and it can vary from micro-turn to micro-turn). Finally, in purely vocal applications, computing the NLG results incrementally does not make much sense as the user's and the system's utterances do not overlap most of the time (Sacks et al., 1974). However, this is an interesting behaviour as far as multimodal applications are concerned.

Building an incremental TTS module can also be very tricky. In order for the synthetic voice to be the most human-like as possible, prosody should be computed accurately and to do so, the sentence's structure and punctuation have to be taken into account. This information is no longer given in the case of incremental TTS or it arrives too late. (Baumann, 2014) proposes a method for coping with the problem.

1.3.4 Existing architectures

Sequential paradigm

A general abstract model of incremental dialogue systems has been introduced in (Schlangen and Skantze, 2011). In this approach, the dialogue chain is maintained and each one of

the five components is transformed into its incremental version. This view of incremental dialogue systems will be referred to as the *sequential paradigm*.

Each module is composed of three parts, the Left Buffer (LB), the Internal State (IS) and the Right Buffer (RB). As described in Section 1.2.2, each module is also characterised by the type of input it processes as well as the type of output it computes. In incremental dialogue, all these data flows have to be divided into small chunks which are called Incremental Units (IU). For example, the audio signal that is given as an input to the ASR module can be divided into 500ms chunks that are processed one by one. Each IU is first added to the LB, then it is taken by the IS for processing and once a result is available, a new IU of a new kind is outputted in the RB. The RB of one module is the LB of the following one in the dialogue chain so the data propagation through the dialogue system is insured.

Because of the ASR instability, new IU in the LB does not necessarily imply that a new IU will be pushed into the RB on top of the ones that already existed there. An example given in (Schlangen and Skantze, 2011) is the following: suppose the user utters the number *forty* which processed incrementally, then first the ASR outputs *four* and then *forty*. As a consequence, the second hypothesis does not complete the first one but it replaces it in the RB. This phenomenon will be discussed in more details in Chapter 4.

Adopting this paradigm is a natural way of enhancing traditional dialogue systems with incremental capabilities. It is interesting from a computational and design point of view as the different tasks are separated. Therefore, one is able to evaluate the different components independently (Baumann and Schlangen, 2011) and have a global view to determine which area still needs improvement.

Multi-layer paradigm

The problem of dialogue management in traditional dialogue systems can be formulated as follows: at each dialogue turn, given the dialogue context (including the last user's utterance), what is the right dialogue act to perform? In the incremental frame, this definition no longer holds as dialogue acts are no longer attached to dialogue turns. Therefore, one way to tackle the problem is to split the dialogue management task in two components, the high-level and the low-level handlers. This paradigm is directly motivated by Austin's, Searl's and Clark's contributions discussed in Section 1.1.1 as the high-level module handles illocutionary acts (the communicative track) whereas the low-level one manages phonetic acts (the meta-communicative track).

As reported in (Lemon et al., 2003), this approach is more in alignment with results in the psycholinguistic field. The phenomena observed at the phonetic level are complex, and the interaction happen on multiple levels, not always following the classical dialogue chain. Having a separate module for handling these phenomena is therefore a more natural paradigm.

Switching from the traditional dialogue management approach to the incremental

one is also a transition from discrete time to continuous time, from a synchronous to an asynchronous processing (Raux and Eskenazi, 2007). The low-level module is continuously (approximated by a high frequency processing in computers) listening to the outside world and waiting for events that might be interesting to communicate to the high-level handler. In that case, the latter returns actions (dialogue acts) and it is the role of the low-level module to choose whether to retrieve them to the user or not as well as choosing the right moment in case it decides to speak.

Finally, starting from a traditional dialogue system, it is easier and more straightforward to transform it into an incremental one if one adopts this paradigm. Adding an extra low-level module to the dialogue manager is enough (Selfridge et al., 2012). At each new incremental input, this module sends the whole partial utterance from the beginning of the current turn to the dialogue manager and gets a response. Based on that and eventually some other features, it decides whether to take the floor or not. As most of the requests sent to the dialogue manager are "fake" as they are not meant to be acted on, they should not affect the dialogue context. Therefore, either multiple instances of the dialogue manager are used, either the dialogue context is saved and restored at each new request, unless the low-level module decides to take the floor (see Chapter 3 for additional explanations).

1.4 Reinforcement Learning

1.4.1 Reinforcement in biology

(To complete and restructure)

Reinforcement Learning (RL) is a sub-field of machine learning where an agent is put into an environment to interact with, and figures out through the process of *trial and error* what the best actions to take are, given a reward function to maximise (Sutton and Barto, 1998) (see Figure 1.3).

1.4.2 Markov Decision Processes

The most common model consists in casting the agent as a Markov Decision Process (MDP) which is a quintuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where:

- $\mathcal{S}$ is the *state space*. At each time step t, the agent is in some state $s_t \in \mathcal{S}$.

- $\mathcal{A}$ is the *action space*. At each time step t, the agent decides to take action $a_t \in \mathcal{A}$.

- $\mathcal{T}$ is the *transition model* where each (s, a, s') in $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is associated with a real number in $[0, 1]$ corresponding to the probability $\mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$. A more compact notation will be used in the following: $\mathcal{T}_{ss'}^a = \mathcal{T}(s, a, s')$.
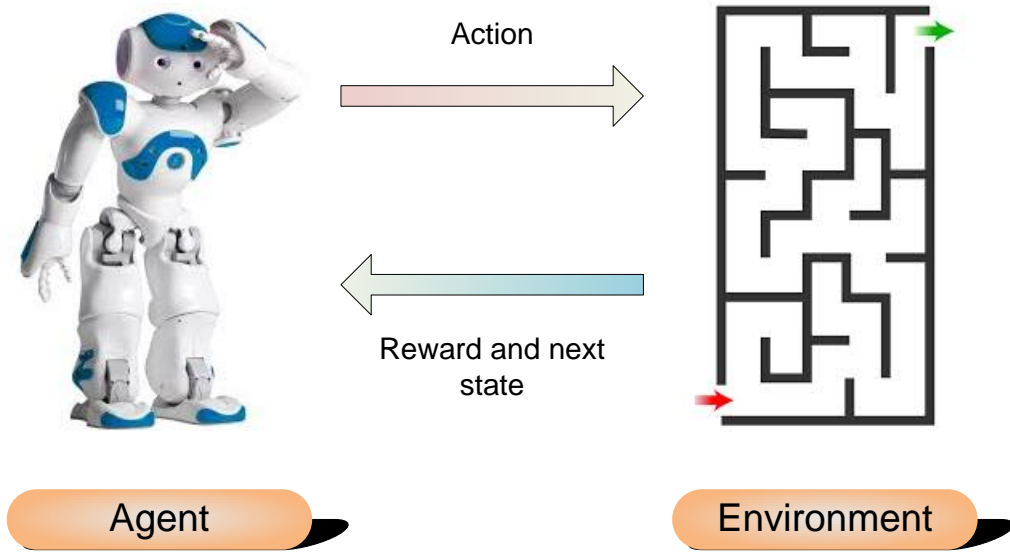
Figure 1.3: *The interaction cycle between the agent and the environment in reinforcement learning*

- $\mathscr{R}$ is the *reward model*. Let r be the immediate reward due to taking action a in state s, then $\mathscr{R}$ is the set of distributions of r for every $(s, a) \in \mathcal{S} \mathrm{x} \mathcal{A}$. The following notation will be used in the rest of this chapter: $\mathscr{R}_{ss'}^{a} = \mathbb{E}[\mathscr{R}(s, a, s')|s, a, s']$.

- $\gamma \in [0, 1[$ is referred to as the *discount factor*. In the RL framework, the aim of the agent is not to maximise the immediate reward but the *expected return*, where the return R_t is defined as follows:

$$
\begin{aligned}
R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... \\
&= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}
\end{aligned}
\tag{1.1}
$$

Therefore, when $\gamma = 0$, the agent maximises the immediate reward only and when γ tends towards 1, the agent maximises the sum of all the future rewards. In other words, the parameter γ controls how far-sighted is the agent in terms of future rewards.

A *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is a mapping between the state space and the action space. An agent is said to follow the policy π when for each time t, it takes the action $a_t = \pi(s_t)$. A policy can also be stochastic, in which case, $\pi(s, a)$ denotes the probability of choosing action a when the agent is at state s. A key aspect of MDPs is the *Markov property*. Being at state s is the only information available to predict the future, and adding information about what happened during previous time steps has no power of prediction. Therefore, given a policy, each state $s \in \mathcal{S}$ is associated with a value $V^\pi(s)$ which is the expected return when at this state and following the policy π afterwards:

$$V^{\pi}(s_t) \;=\; \mathbb{E}[R_t | s_t, \pi] \tag{1.2}$$

Another interesting quantity is the expected return knowing the current state but also the next action, after which π is followed. This is referred to as the Q-function:

$$Q^{\pi}(s_t, a_t) \;=\; \mathbb{E}[R_t | s_t, a_t, \pi] \tag{1.3}$$

Given the definition of R_t, one can notice that

$$
\begin{aligned}
V^{\pi}(s_t) \;&=\; \mathbb{E}[R_t | s_t, \pi] \\
&=\; \mathbb{E}\Big[r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{(t+1)+k+1} \Big| s_t, \pi\Big] \\
&=\; \mathbb{E}[r_t + \gamma R_{t+1} | s_t, \pi] \\
&=\; \mathbb{E}[r_t + \gamma \mathbb{E}[R_{t+1} | s_{t+1}, \pi] | s_t, \pi] \\
&=\; \mathbb{E}[r_t + \gamma V^{\pi}(s_{t+1}) | s_t, \pi]
\end{aligned}
\tag{1.4}
$$

This is known as the Bellman equation for V^{π} and it can also be written for the Q-function, as follows

$$Q^{\pi}(s_t, a_t) \;=\; \mathbb{E}[r_t + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1})) | s_t, a_t, \pi] \tag{1.5}$$

1.4.3 Reinforcement Learning

A natural question that can be asked at this point is: how are these values computed? In reinforcement learning, this is known as the *evaluation problem*. The transition model $\mathscr{T}$ and the reward model $\mathscr{R}$ are the elements that define the dynamics of the MDP. If they are known, V^{π} can be directly computed:

$$
\begin{aligned}
V^{\pi}(s) \;&=\; \mathbb{E}[R_t | s_t = s, \pi] \\
&=\; \sum_{a \in \mathcal{A}} \pi(s, a) \mathbb{E}[R_t | s_t = s, a_t = a, \pi] \\
&=\; \sum_{a \in \mathcal{A}} \pi(s, a) \mathbb{E}[r_t + \gamma R_{t+1} | s_t = s, a_t = a, \pi] \\
&=\; \sum_{a \in \mathcal{A}} \pi(s, a) \sum_{s' \in \mathcal{S}} \mathscr{T}^a_{ss'} (\mathscr{R}^a_{ss'} + \gamma \mathbb{E}[R_{t+1} | s_{t+1} = s', \pi]) \\
&=\; \sum_{a \in \mathcal{A}} \pi(s, a) \sum_{s' \in \mathcal{S}} \mathscr{T}^a_{ss'} (\mathscr{R}^a_{ss'} + \gamma V^{\pi}(s'))
\end{aligned}
\tag{1.6}
$$

It is possible to define an order over the policies. Saying that π_1 is better that π_2 means that for all the states s, $V^{\pi_1}(s) \geq V^{\pi_2}(s)$. It can be shown that there exists at least one policy that is better than all the others: it is called the *optimal policy* (π^*). To simplify the notations, V^{π^*} will be referred to as V^* and it is defined as

$$\forall s \in \mathcal{S},\ V^*(s)\ =\ \max_{\pi} V^{\pi}(s) \tag{1.7}$$

Similarly, one can define Q^* as

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A},\ Q^*(s,a)\ =\ \max_{\pi} Q^{\pi}(s,a) \tag{1.8}$$

The aim of reinforcement learning is to learn the optimal policy. Similarly to what has been shown for V^{π}, if the transition and the reward models are known, the Bellman equation corresponding to V^* (called the *Bellman optimality equation*) can be written with respect to these models (similarly to 1.6):

$$V^*(s)\ =\ \max_{a} \sum_{s' \in \mathcal{S}} \mathcal{T}^a_{ss'}(\mathcal{R}^a_{ss'} + \gamma V^*(s')) \tag{1.9}$$

A similar form can be also be shown about the Q-function

$$Q^*(s,a)\ =\ \sum_{s' \in \mathcal{S}} \mathcal{T}^a_{ss'}(\mathcal{R}^a_{ss'} + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a')) \tag{1.10}$$

A set of *Dynamic Programming* methods exist in order to efficiently solve these kinds of equations and come up with the optimal policy given the transition and the reward model (knowing Q^* implies knowing π^* as the latter is the greedy policy with respect to the former Q-function, in the sense that $\pi^*(s) = \operatorname*{argmax}_{a} Q^*(s,a)$). However, even though this kind of approaches are theoretically interesting, they only have a few practical applications as most of the times, $\mathcal{T}$ and $\mathcal{R}$ are unknown. The agent learns directly from interacting with the environment (*model-free* approach).

It is possible to try to learn $\mathcal{T}$ and $\mathcal{R}$ first and then applying a *model-based* algorithm to figure out the optimal policy. Nevertheless, this is not necessary as most algorithms compute the optimal policy by directly estimating the Q-function. This can be done in a straightforward fashion by running several episodes[1], computing the returns for each state-action couple and for each episode, then using the mean return over all the

---

[1]To keep things simple in this introduction to reinforcement learning, the MDP is considered to eventually stop.

episodes as an estimate of V^π or Q^π. Algorithms using this kind of approach belong to the category of *Monte-Carlo methods*.

However, as the agent interacts with the environment, it encounters a similar dilemma to the one faced in the bandit problem (Berry and Fristedt, 1985; Bubeck and Cesa-Bianchi, 2012): how to manage the trade-off between *exploration* and *exploitation*. While being at a state $s \in \mathcal{S}$, the agent can choose one action among many. Let us say that the Q-function is initialised as a zero function. Therefore, at the beginning the agent has no preference and selects a random action. If this yields a positive reward, then the agent has the choice between these two options to make the next decision:

1. Making the same decision again as it already knows that it is likely to generate a positive reward.

2. Picking another action because it may yield an even greater reward.

In the first case, the agent is exploiting its current knowledge of the environment whereas in the second case, it is said to be exploring as it is increasing its knowledge about the environment (with the risk of generating low or negative rewards in the meanwhile). Because rewards are stochastic, it is not obvious to determine whether sufficient data is available to trust our estimates and start exploiting most of the time. This is a difficult problem and a simple way to deal with it is to use the ϵ-greedy approach, where the agent chooses a random action with a probability of ϵ and sticks to the greedy action (with respect to the current estimated Q-function) the rest of the time. Nevertheless, more robust solutions have already been suggested like (Auer et al., 2002) for the bandit problem and (Auer and Ortner, 2005) for reinforcement learning.

Reinforcement learning algorithms keep evaluating the current policy and at the same time, altering that policy in order to improve it. A naive approach would be to fix the current policy and to perform as many evaluation iterations as necessary in order to gain a certain confidence over the estimations of V or Q and then to derive a new policy to follow, given these values. This is known as *Policy Iteration* but this is not the most efficient way to proceed (so many iterations are needed). In fact, performing only one evaluation iteration before the next policy improvement step can be shown to be enough, keeping the convergence guarantees. This is referred to as *Value Iteration*. Also, the notion of iteration can be viewed differently given the approach and the algorithm at hand. In order to refer to the general idea of intertwining evaluation and control, the expression *General Policy Iteration* (GPI) is used.

In fact, it is also possible to evaluate V or Q in an even more fine-grained manner. Instead, of waiting until the end of the episode to update these values, it is possible to do it after each new decision. That is what *Temporal-difference (TD) methods* do. In comparison with the Monte-Carlo approach, the new sample for $V^\pi(s)$ or $Q^\pi(s, a)$ is no longer the real return obtained in the episode but an estimated one using the Bellman equation. In the case of the *sarsa* algorithm[2], the Q-function is updated as follows[3] (α_t

---

[2]See (Sutton and Barto, 1998) for the algorithm description.

[3]At this point, V will no longer be used, as Q is the most commonly used in reality. V is mostly used for pedagogical purposes.

being a decreasing parameter with time):

$$Q_t(s_t, a_t) \quad = \quad Q_t(s_t, a_t) + \alpha_t[r_t + \gamma Q_t(s_{t+1}, \pi_t(s_{t+1})) - Q_t(s_t, a_t)] \qquad (1.11)$$

It is important to notice that a_{t+1} is the action chosen by following the current estimated policy derived from Q (ϵ-greedy for example) and which will be actually followed in the next step. The sarsa algorithm is therefore called an *on-policy* algorithm. These conditions can be relaxed giving birth to another category of algorithms, the *off-policy* ones. The most famous is *Q-Learning*[4] (Watkins, 1989) where the Q-function is updated as follows:

$$Q_t(s_t, a_t) \quad = \quad Q_t(s_t, a_t) + \alpha_t[r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)] \qquad (1.12)$$

Here, the policy used for evaluation is not necessarily the one that is followed.

1.5 Reinforcement learning in spoken dialogue systems

1.5.1 In the litterature

(Take some information from Layla's thesis to talk about summary states and industrial SDSs, in addition to the following paragraph...).

Reinforcement learning has been first applied to dialogue systems in (**?**) and since then, it has been the leading machine learning framework in the field. The dialogue state at time t is generally determined by the history of dialogue acts since the beginning of the dialogue. At each turn, the set of actions is made of all the possible answers at that time. Partially Observable Markov Decision Processes (POMDPs) (Williams and Young, 2007) can also be used. In this framework, the dialogue state is replaced by a distribution over all possible states which is a more natural way of modeling uncertainty, however, they are more complex and more difficult to scale (Lemon and Pietquin, 2007).

1.5.2 Spoken dialogue systems at Orange and the LIA

1.5.3 Dialogue simulation

A couple of decades ago, with the development of the dialogue systems research field, the need for evaluation means in order to assess their quality started getting more and more important. Therefore, researchers turn to user simulation methods (also referred to as user modeling). In (Eckert et al., 1997), some of the advantages of these techniques

---

[4]See (Sutton and Barto, 1998) for the algorithm description.

are depicted: reduced cost with automatic evaluation of a large number of dialogues, less error risk, easy modeling of different user populations, possibility of using the same user model across different concurrent dialogue systems for comparison and providing a tool to quickly generate corpora for machine learning techniques at a low cost. Nevertheless, the authors recognise that user simulation cannot totally replace interactions with real users in the process of designing reliable dialogue systems: *we believe that tests with human users are still vital for verifying the simulation models*.

Simulating users accurately is a challenging task as their behaviours vary considerably from one person to another and the same user can change her preferences over time (concept-drift) (Schatzmann et al., 2006). Evaluating a user simulator and whether it handles such variability or not is a research track in itself (Pietquin and Hastie, 2013) and the qualities required are of different kinds. The trained user simulator should be consistent with the data that has been used for the training and the sequence of dialogue acts generated should be coherent. In addition, when it is used in turn to train a data-driven dialogue strategy, the quality of the latter is also an evaluation criteria. Also, it is important that the results obtained in simulation give strong indications about the behaviours with real users.

User simulation is useful during the conception phase of a dialogue system. However, training the simulator from data needs the dialogue system to be conceived already. Therefore, trying to come up with a simple model with only a few parameters is not always a bad idea as it has been proven to achieve good results as well (Schatzmann et al., 2007).

User simulator is also quite similar to the dialogue management task. As a consequence, it is legitimate to ask the following question: why not use reinforcement learning to train user simulators? The answer is that in the case of dialogue management, it is easier to come up with a reasonable reward function: task completion, dialogue duration, subjective evaluation...etc... When it comes to user simulation, the objective function is how well a real user is imitated which is impossible to evaluate. Fortunately, there exists a framework where the reward function is automatically inferred from data which is particularly useful here: Inverse Reinforcement Learning (Chandramohan et al., 2011; Asri et al., 2012).

When it comes to incremental dialogue systems, the only existing user simulator in our knowledge is the one described in (Selfridge and Heeman, 2012). Its state is updated every 10 ms. However, the *ASR instability* phenomenon is not replicated, that is to say that the ASR hypothesis construction is monotonous whereas in reality, it is the heart of the problem. When a new audio signal increment is heard by the ASR, the output can be partially or totally modified. In this simulator, only the simple case where a new increment is added to the output is modeled.

1.6 Reinforcement learning in incremental dialogue systems

In the field of incremental dialogue and turn-taking management, supervised learning is more common. The main problem tackled by researchers is the identification of the exact moments where the system should take the floor in order to achieve smooth turn-taking (Raux and Eskenazi, 2008; Gravano and Hirschberg, 2011; Meena et al., 2013). Binary classifiers are used and the features they are fed are of different natures: lexical, semantic, prosodic...etc...However, a few papers tackled this problem by using reinforcement learning.

(Jonsdottir et al., 2008) used reinforcement learning while considering prosodic features only. Backchanneling for example can be performed by humans independently from the meaning. The cost function (negative reward) is taken as gaps and overlaps, hence following Sack's principle discussed in Section 1.1.2.

(Dethlefs et al., 2012) adopted a complementary approach where only the semantic content of the user's utterance is taken into account (hierarchical reinforcement learning is used). In human conversation, it is more likely for the listener to react right after a relevant information. Similarly, in the case of a restaurant finding spoken dialogue system, the system should react right after understanding the restaurant's type or price range. In this work, the information pertinence is measured by the Information Density (ID). Therefore, the more the ID is high during system actions, the more reward it gets.

Instead of trying to minimise gaps and overlaps, the reward function can be designed in a way to optimise dialogue duration and task completion like it is the case in (Selfridge and Heeman, 2010). The system in this paper learns optimal initial turn-taking, in the sense that when a silence is detected, the dialogue participant that has the most relevant thing to say takes the floor first. Like in the previous paper, only semantic features are considered.

A third approach to optimise turn-taking in spoken dialogue systems is to directly try to imitate human behaviours. In (Kim and Banchs, 2014) Inverse Reinforcement Learning is used to infer a reward function directly from user trajectories in a gathered dialogue corpus. Therefore, the reward function automatically incorporates objective and subjective dialogue quality criteria. The authors have made the choice not to consider lexical and semantic features, but rather to limit their work to timing and prosody signals.

(Conlude and recall the motivations).

Part II

Contributions

Chapter 2

Turn-taking taxonomy

2.1 Introduction

In this chapter, the following question is tackled: what is turn-taking? In (Raux and Eskenazi, 2009), four basic turn-taking transitions are introduced: *the turn transitions with gap*, *the turn transitions with overlap*, *the failed interruptions* and *the time outs*. A *turn-taking labeling scheme* has been proposed in (Gravano and Hirschberg, 2011) (which extends the original *classification of interruptions and smooth speaker-switches* introduced in (Beattie, 1982)). This classification is richer as the meaning of the involved turn-taking behaviours is taken into account. Here, the aim is to go even further by using several levels of analysis in order to introduce a new taxonomy of turn-taking phenomena (Khouzaimi et al., 2015). It is aimed to provide a better grasp of the concept of turn-taking by breaking it into small manageable pieces that can be implemented and studied separately.

In Chapter 1, the reader is given some clues and some previous work references in order to build a first intuition of what turn-taking is. Here, an analysis of turn-taking in human conversation is performed. It is aimed to provide an answer to the four following questions:

1. What phenomena characterise turn-taking in human conversation?

2. How can they be classified in order to clearly identify the similarities and the differences between them?

3. What are the general categories that emerge from the general picture drawn by this classification?

4. What phenomena are worth implementing in dialogue systems and why?

Each element of the proposed taxonomy will be referred to as a *turn-taking phenomenon* (TTP). The analysis levels laid in the philosophy of language will be used here while discussing the taxonomy: the locutionary, the illocutionary and the perlocutionary paradigms. Nevertheless, the locutionary act concept is subject to a few disagree-

ments between J. L. Austin and his successor, J. R. Searle. Looking deeper into a locutionary act, it can be broken into three sub-levels: the *phonetic*, the *phatic* and the *rhetic* which correspond to the verbal, the syntactic and the semantic levels. In (Searle, 1968), the author argues that there is no distinction possible between the rhetic level and the illocutionary one and therefore refuses the more general distinction between locutionary and illocutionary level. As a result, he suggests to adopt the four layer structure composed of: *phonetic acts*, *phatic acts*, *propositional acts* (the act of expressing the proposition) and *illocutionary acts*. Nevertheless, the philosophical subtleties brought by these considerations are beyond the scope of this thesis and the objective in this chapter is to provide pragmatic criteria that will make it possible to distinguish the several TTP at hand. Therefore, only the three following analysis levels have been taken from this theory of language (as they are enough for this analysis): the phonetic, the illocutionary and the perlocutionary levels.

Recall that at the perlocutionary level, the impact that a dialogue act is aimed to have is considered, like convincing, congratulating or insulting for example. Here, an extra dimension is also needed: what is the motivation behind a dialogue act? In the taxonomy introduced in this chapter, some TTPs are exactly the same if viewed as phonetic, illocutionary and perlocutionary dialogue acts, but the reason why they are performed are different. Making this distinction are interesting from a computational point of view as it is directly correlated to the set of features that are considered by the system in order to make turn-taking decisions.

2.2 Taxonomy presentation

Let us consider the three following dialogue situations.

Dialogue 1

ALBERT: I would like to try some exotic destination this summer where I can ...

BETTY: ... Have you ever been to India?

Dialogue 2

ALBERT: First you put the meat in the oven ...

BETTY: ...aha...

ALBERT: ...then you start preparing the salad...

Dialogue 3

ALBERT: What time is it please?

BETTY: It is half past two.

In all the dialogues, Albert initially has the floor and then Betty performs a dialogue act. In dialogue 1 and 2, he does not wait for her to finish her utterance before doing so, unlike in the last dialogue. Therefore, Betty can choose the timing of his intervention at different stages in the progression of Albert's utterance. The first criterion used in the taxonomy introduced here corresponds this decision. Moreover, in dialogues 1 and 3, Betty utters a complete sentence unlike in dialogue 2. The second criterion is aimed to make the distinction between these kind of behaviours performed by Betty.

More formally, turn-taking in dialogue refers to the act of taking the floor by one participant (Betty in the previous examples), here called the Taker (T). Two cases can be distinguished; either the other participant, here called the Holder (H), is already speaking or not (the denomination Holder is more adapted to the case where it has the floor, but it is kept as a convention for the other case).

The taxonomy introduced here is based on two dimensions:

1. **The quantity of information that H has already injected in the dialogue:** This measures how early in H's utterance T chooses to perform her dialogue act.

2. **The quantity of information that T tries to inject by taking the floor:** T's dialogue act can consist on some implicit reaction (gestures, sounds like *aha*), a complete utterance or something in between.

The different levels of information for each dimension are described on Table 2.1.

H_NONE	No information given
H_FAIL	Failed trial
H_INCOHERENT	Incoherent information
H_INCOMPLETE	Incomplete information
H_SUFFICIENT	Sufficient information
H_COMPLETE	Complete utterance
T_REF_IMPL	Implicit ref. to H's utterance
T_REF_RAW	Raw ref. to H's utterance
T_REF_INTERP	Reference with interpretation
T_MOVE	Dialogue move (with improvement)

Table 2.1: *Taxonomy labels*

Table 2.2 describes the taxonomy where turn-taking phenomena (TTP) are depicted. The rows correspond to the levels of information added by H and the columns to the information that T tries to add. In order to describe each one of them in detail, they are discussed row by row.

H_NONE: H does not have the floor, therefore, T takes the floor for the first time in the dialogue. This can be done implicitly by performing some gesture to catch H's attention or by clearing her throat for instance (FLOOR_TAKING_IMPL). On the other hand, she can start speaking normally (FLOOR_TAKING_EXPL).

41

	T_REF_IMPL	T_REF_RAW	T_REF_INTERP	T_MOVE
H_NONE	FLOOR_TAKING_IMPL			INIT_DIALOGUE
H_FAIL	FAIL_IMPL	FAIL_RAW	FAIL_INTERP	
H_INCOHERENCE	INCOHERENCE_IMPL	INCOHERENCE_RAW	INCOHERENCE_INTERP	
H_INCOMPLETE	BACKCHANNEL	FEEDBACK_RAW	FEEDBACK_INTERP	
H_SUFFICIENT	REF_IMPL	REF_RAW	REF_INTERP	BARGE_IN_RESP
H_COMPLETE	REKINDLE			END_POINT

Table 2.2: Turn-taking phenomena taxonomy. The rows/columns correspond to the levels of information added by the floor holder/taker.

H_FAIL: H takes the floor for long enough to deliver a message (or at least a chunk of information) but T does not understand anything. This can be due to noise or to the fact that the words and expressions are unknown by T (other language, unknown cultural reference, unknown vocabulary...). T can interrupt H before the end of his utterance as she estimates that letting him finish it is useless. This can be done implicitly (FAIL_IMPL) using a facial expression (frowning), a gesture or uttering a sound:

H: Cada hora que paso aquí...

T: ...what?

It can also be done by explicitly uttering that H's utterance is not clear so far (FAIL_-RAW):

H: <noise> has been <noise> from...

T: ...sorry, I can't hear you very well! What did you say?

Finally, T can interrupt H by trying to provide a justification to the fact that H needs to repeat, reformulate or add complementary information in his sentence (FAIL_INTERP). For example:

H: Freddy was at the concert and ...

T: ...who is Freddy?

H_INCOHERENCE: T understands H's message and detects an incoherence in it, or between that message and the dialogue context. H can make a mistake like *I went swimming from 10 am until 9 am*, or, *First, go to Los Angeles, then go south to San Francisco...* He can also be unaware of the dialogue context: *You should take metro line A...* when metro line A is closed that day. Again, this can be done implicitly (INCOHERENCE_IMPL) by adopting the same behaviours as in the case of H_FAIL, or explicitly (INCOHERENCE_RAW).

H: Investing in risk-free instruments like stocks is one of the ...

T: ...that is nonsense.

T can also explain the reasons she thinks this is not coherent (INCOHERENCE_-INTERP):

H: I will visit you on Sunday and then ...

T: ...but you are supposed to be traveling by then!

H_INCOMPLETE: H's utterance is still incomplete (and H is still holding the floor) but all the information given so far is coherent. T can perform a backchannel by nodding her head for example or by saying *Aha* or *Ok* for example (BACKCHANNEL). This gives H a signal that he is being understood and followed, thus encouraging him to keep on speaking. T can also choose to repeat a part of H's sentence for confirmation (FEEDBACK_RAW). If this part is correct, H continues to speak normally (or sometimes explicitly confirms by adding a *yes* to his sentence), otherwise he declares that he disagrees with T's feedback. Here is an illustration of this mechanism taken from (Khouzaimi et al., 2014):

H: 01 45

T: 01 45

H: 65 79

T: 67 79

H: No, 65 79

T: Sorry, 65 79

H: 98

T: 98

H: ...

T: The dictated number is: 01 45 65 79 98. Is that correct?

H: Yes

Another kind of feedback is by adding some related information to H's incomplete utterance (FEEDBACK_INTERP), for example:

H: I went to see the football game yesterday...

T: ...yeah, disappointing

H: ...with a friend, but we did not stay until the end.

H_SUFFICIENT: H has not finished talking, yet, all the information that T needs to answer has been conveyed. If H is listing a few options, T can perform a gesture meaning that she is interested in the last option uttered (REF_IMPL). She can also do it explicitly (REF_RAW) (see (El Asri et al., 2014b)):

H: Maybe we can for an appointment on Monday afternoon?...Tuesday morning?...Wednesday afternoon?...

43

T: Ok. Fine.

T can also add comments related to her choice, once selecting an option (REF_-INTERP):

H: We have apple juice...tomato juice...

T: Oh Yeah! That is my favorite, plus, my doctor advised me to have some every day.

In the case of goal-oriented dialogue, H keeps talking even though he conveyed all the necessary information for T to formulate an answer. T can choose to interrupt him (BARGE_IN_RESP) thus making the dialogue shorter (this can be viewed as a rude move in some cases):

H: I want to book a six person table tomorrow at 6 please, I hope it is possible since I have ...

T: Sure, no problem. Can I have your name please?

H_COMPLETE: H has finished his utterance. If T thinks that some more information needs to be provided, she can perform a gesture or adopt a facial expression to communicate that (REKINDLE), making H take the floor again and provide further information. This can also be done explicitly and it will be considered as a new dialogue turn, as well as T providing new information to make the dialogue progress (END_POINT). The latter is the most intuitive TTP that people have in mind when trying to model turn-taking.

H: How many friends of yours are coming with us tomorrow?

T: Two, hopefully.

2.3   Discussion

This taxonomy is aimed to clarify the notion of turn-taking. In human-human conversation, this translates into a rich set of behaviours that are depicted and classified given two criteria. Compared to existing classifications of turn-taking behaviours, an important part is given to the semantic content of H's and T's utterances (and other cues like gestures and facial expressions) as well as the reasons that pushed T to take the floor given this information.

As far as replicating TTPs in human-machine interactions is concerned, a big part of research in incremental dialogue systems and turn-taking optimisation has mainly focused on endpoint detection (Raux and Eskenazi, 2008) and smooth turn-taking. Therefore, their objective is to replicate the phenomenon labeled here as BARGE_IN_RESP. Some other studies focus on backchanneling and feedback, often neglecting the semantic part of the dialogue participants utterances and focusing exclusively on prosody and acoustic features (Baumann, 2008; **?**).

44

The identified TTP can be classified in five categories (referenced by different colors in Table 2.2):

1. Dialogue initialisation (gray)

2. Negative feedback (red)

3. Positive feedback (blue)

4. Reference (yellow)

5. Ordered complete dialogue acts (green)

In the following, each category is discussed separately. Before starting the analysis, let us briefly recall the four different levels of analysis used here (see Chapter 1 for a more complete review):

1. **Phonetic level:** The dialogue acts are considered as a sequence of sounds. This is what a Voice Activity Detection (VAD) module would detect.

2. **Illocutionary level:** The message that can be extracted from this sequence of sound is the focus here.

3. **Perlocutionary level:** Refers to the effect that the dialogue act is supposed to have on the listener.

4. **Reason behind the dialogue act:** The reason that pushed T to perform this dialogue act.

2.3.1 Dialogue initialisation

Two TTPs constitute this category: FLOOR_TAKING_IMPL and INIT_DIALOGUE. They should be distinguished from REKINDLE as they take place at the very beginning of the dialogue or when the dialogue participants stopped interaction for a long while (so that it is legitimate to consider that they are engaging in a new interaction). Viewed as phonetic dialogue acts, they are different most of the time. The first one involves implicit gestures and short sounds or words whereas in the second one, T makes brings some new information to the table. However, in some cases, it is possible to inject new information in very short sentences, therefore the length of the sentence cannot be used as a criterion to distinguish between these two TTPs. As an illustration, imagine that you observe people that are interacting using a language that is unknown to you. Imagine, that they are silent and suddenly one of them utters a short sound. In that case, it is not obvious whether he just called his interlocutor's name or whether he actually uttered some new piece of information.

Considering the illocutionary level, FLOOR_TAKING_IMPL contains no information in itself apart from the fact that T wants to take the floor. INIT_DIALOGUE, on the other hand, has a meaning as it contains new information. This is the main distinction between the two phenomena. As a consequence, viewed as a perlocutionary act, INIT_DIALOGUE plays a double role: making H aware that T is starting an interaction

(shared with FLOOR_TAKING_IMPL) and adding new information at the same time. Finally, the reason why H performs FLOOR_TAKING_IMPL is the the desire to start a new interaction. INIT_DIALOGUE is also performed for the same reason but in addition, it is also aimed to announce the conversation subject and/or to make H aware of some new piece of information.

2.3.2 Negative feedback

Negative feedback is communicated through one of the six following phenomena: FAIL-_IMPL, FAIL_RAW, FAIL_INTERP, INCOHERENCE_IMPL, INCOHERENCE_RAW and INCOHERENCE_INTERP. They all suggest that both participants have to take a step back in order to clarify or to correct something in the dialogue. From a phonetic point of view, like in 2.3.1, there is no rigourous distinction between these TTPs (unless the implicit ones are only gestures or facial expressions), even though the implicit ones are supposed to be shorter than the implicit ones, which in turn are supposed to be shorter than the interpreted ones.

It is interesting to notice that, viewed as a phonetic and an illocutionary dialogue act, FAIL_IMPL and INCOHERENCE_IMPL are the same or at least extremely hard to distinguish. This is also true for FAIL_INTERP and INCOHERENCE_INTERP. These dialogue acts translate into exactly the same signal sent by T, and only the dialogue context makes it possible to separate them. As perlocutionary act they are slightly different as they are both make H stop and take a step back in the dialogue. However, they are different as in the case of FAIL TTPs, the goal is to make H repeat the same sentence again (or rephrases it while keeping the same meaning), whereas in the case of INCOHERENCE TTPs, T want H to change his sentence and its meaning because it is problematic.

Actually, the real difference between FAIL and INCOHERENCE TTPs comes from the fourth level that has been added to the analysis: the motivation behind behaving as such (these kind of differences is what motivated adding this dimension of analysis). As said earlier, the behaviour can be identical between the two categories (frowning, or saying *What?* for example), but the core different between them comes from the fact that what pushes T to perform a FAIL TTP is the fact that she does not understand what has been said by H so far, and she does not want to lose track of the conversation, whereas in the case of an INCOHERENCE, she feels the need to signal a problem.

2.3.3 Positive feedback

BACKCHANNEL, FEEDBACK_RAW, FEEDBACK_INTERP and REKINDLE are aimed to give H a positive feedback in the sense that, unlike negative feedback, he is encouraged to keep the floor and to keep injecting new information. BACKCHANNEL and REKINDLE are generally shorter from a phonetic point of view (they can be also be gestures) but they are different as BACKCHANNEL involves an overlap whereas

REKINDLE is performed after H releases the floor. FEEDBACK_RAW and FEED-BACK_INTERP are usually longer but there is no difference between them at this level of analysis.

As illocutionary acts, BACKCHANNEL, FEEDBACK_RAW and REKINDLE are all close to the dialogue act *I understand what you said so far, please continue* (with a few subtle differences, though). On the other hand, FEEDBACK_INTERP is richer as new information is injected. At the perlocutionary level, T wants to have a double effect on H:

1. Reassure him that his message has been understood so far.

2. Encouraging him to go on and add more information.

Finally, when considering the reasons that pushed T to perform these TTPs, REKIN-DLE detaches itself as it is the only one where T is surprised that H's utterance has already stopped. As a consequence, she feels the urge to ask for more.

2.3.4 Reference

REF_IMPL, REF_RAW and REF_INTERP are the TTPs that constitute this group. Again, the phonetic analysis does not provide interesting insights apart from the fact that REF_IMPL can be a gesture or a shorter speech act than REF_RAW and REF_INTERP. This category is interesting from an illocutionary point of view as the message that T tries to send is not present in her utterance but in H's one.

From a perlocutionary perspective, these TTPs are aimed to make H stop and understand T's answer from his own sentence. Finally, what pushes T to act this way is to avoid repetition and to be more efficient.

2.3.5 Ordered dialogue acts

In this last section, BARGE_IN_RESP and END_POINT are discussed. The simplest way of viewing dialogue is by adopting the walkie-talkie paradigm. Time is shared between participants in a sequential way where each one of them takes the floor and then releases it for the other to speak. As described previously, this corresponds to the END_POINT phenomenon. Viewed as a phonetic act, it is characterised by the absence of overlap (or very small overlaps) and even a gap most of the time. On the other hand, no gaps are involved in BARGE_IN_RESP and overlaps are frequently observed.

There is no difference between the two phenomena at the illocutionary level, however, considered as perlocutionary acts, T tries to inject new information in both cases but BARGE_IN_RESP comes with the additional intent of making H stop talking. T is pushed to act as such whenever she thinks that she has enough information to start uttering her next dialogue act. Therefore, the motivation behind such a behaviour is increasing efficiency by suppressing an unnecessary part of H's utterance hence gaining time. However, in some situations, barge-in cannot be performed either because of real

constraints (a real walkie-talkie conversation for example) or because of social codes (politeness, timing allowed during an official meeting or a hearing...).

2.3.6 Synthesis

In tables 2.3 and 2.4, the previous analysis is synthesised in the form of a table. A phonetic profile is associated with each phenomenon (it is not accurate nor it is always respected, it is only aimed to give a general idea about how the TTP takes place in time between H and T), as well as a description of the illocutionary and perlocutionary acts. The elements motivating each phenomenon also appear in the table (x and y are variables that are used to refer to the content of the dialogue acts).

2.4 Turn-taking phenomena in dialogue systems

INIT_DIALOGUE is a TTP that is involved in every dialogue system, including traditional ones. There are two ways of initialising the dialogue, the user initiative one (the user starts speaking first like this is the case for Siri for instance) and the system initiative way (the system delivers an initial prompt, when calling an IVR for example). END_POINT is also necessary for any kind of dialogue, however, the way the dialogue participants exchange turns is not always the same given the situation at hand. Humans are very good at detecting end of utterance clues beforehand, making T able to anticipate the right moment to take the floor hence achieving smooth turn-taking. Traditional dialogue systems, on the other hand, rely on long enough silences as markers of end of turn. A research thread is dedicated to studying methods of reducing these silences by considering different clues (prosodic, lexical and semantic) aiming for smoother turn exchange (Raux and Eskenazi, 2008; Gravano and Hirschberg, 2011). However, even though they have been under study for many years, there is still room for improvement. In the rest of this section, the remaining TTPs are discussed (that can be replicated in incremental dialogue systems only) from an implementation point of view. Our goal is to come up with a list of TTPs that are the most likely to improve task-oriented dialogue. Similarly, REKINDLE is a dialogue management strategy that can be implemented in any traditional dialogue system and as a consequence, it will not be considered here.

Before leading this discussion, it is important to notice that each TTP has two symmetric versions when it comes to human-machine dialogue: the one where H is the user and T is the machine and the opposite case. Here, the goal is to study system decisions therefore only one side is mainly studied (even though a TTP is implemented from both sides). In order for both cases to be implemented, the incremental dialogue system at hand should always be listening to the user, even though it has the floor (hence being able to be interrupted). As a technical side note, current incremental dialogue systems are used with a headphone for that reason: as the system keeps listening all the time, it is a convenient way to prevent it from hearing itself while speaking (considering its own sentence as a user input). In order to make them useful outside of labs, in more

realistic situations, it is necessary to build algorithms that suppress the TTS result from the ASR input before feeding it to the latter (which raises a new challenge as it must also be done incrementally). Moreover, in this thesis, the focus is on the vocal modality. Therefore, no TTP based on gestures is considered for implementation.

Implementing a mechanism that mimics the FAIL TTPs is an interesting idea to explore. Users frequently use off-domain words and expression (Ghigi et al., 2014) and they are also ofter misunderstood by the ASR. As a consequence, making the system barge-in when it does not understand the user's partial utterance might have a positive impact on the dialogue efficiency. It might be less tiresome for the user as she wouldn't have to repeat her whole sentence several times. Moreover, this reduces the dialogue duration. FAIL_RAW is not easy to replicate by a machine and it is most of the time performed with gestures and facial expressions at the same time. FAIL_INTERP is not easy to implement neither as giving the accurate reason why it did not manage to understand the user's utterance so far is not an obvious task. Implementing FAIL_RAW, on the other hand, is much more realistic: when the systems has no clue about the user's utterance after a long enough period of time, it simply declares that fact in a straightforward fashion.

In some cases, the user is likely to utter a sentence that is not coherent for the system or that is in contradiction with some data accessible by the latter (like when trying to buy a movie ticket when all the seats are already sold). By definition, the INCOHERENCE TTPs can help manage this case in a more efficient way. For the same reasons as FAIL_RAW, INCOHERENCE_RAW is not easy to implement. Unlike FAIL TTPs, it is not natural and more difficult to declare an incoherence without explaining the underlying reasons. Therefore, INCOHERENCE_INTERP is the most interesting TTP to implement.

BACKCHANNEL has already been implemented in a few incremental dialogue systems (Meena et al., 2013; Hastie et al., 2013) with the aim of increasing its grounding capacities and its naturalness. This thesis focuses more on the efficiency aspect of incremental dialogue than the human-likeness side of the problem. These are somehow correlated, but as this is not a TTP that directly makes the dialogue more efficient (by preventing and fixing errors), it will not be implemented. Moreover, as the first step of the approach followed here is to use simulated dialogues, it is hard to evaluate this TTP in such conditions. On the contrary, FEEDBACK_RAW provides a concrete opportunity to correct errors. When T tries to repeat a part of T's utterance and she succeeds, this gives H a proof that T heard his sentence (even though, this does not necessarily mean that T has well understood the message). If she fails, it is also interesting as H can repeat or reformulate his utterance, hence avoiding a desynchronisation. This is clearly interesting to be implemented in a dialogue system, yet, it can be very challenging. The involved turn-taking mechanism is difficult to manage in the sense that the user should not interpret the system's intervention as a barge-in hence being interrupted. Moreover, the system should be able to recognise whether the user ignored the feedback or tried to correct its content. Therefore, this TTP has been implemented in simulation only. Finally, FEEDBACK_INTERP requires high NLP capabilities and access to an important knowledge base. Therefore, it has not been implemented here.

In (El Asri et al., 2014b), REF_RAW has been implemented from the user's point of view: the system enumerates a list of alternatives and the user barges-in to select one of them. This has been shown to significantly increase the dialogue quality. However, implementing it requires changing the dialogue management strategy whereas, as discussed in Chapter 3, this thesis focuses on the impact of adding a turn-taking layer on top of pre-existing dialogue management strategies. As a consequence, REF_IMPL, REF_RAW and REF_INTERP are not studied here.

Finally, BARGE_IN_RESP is clearly worth implementing from both sides. From the system's perspective, taking the floor as soon as it has enough information to do so can directly increase dialogue efficiency by reducing dialogue duration but also indirectly by preventing the user from adding new misleading information (Ghigi et al., 2014). From the user's point of view, being able to take the floor before the end of the system's utterance can make the dialogue less tiresome. This is especially true for users that are familiar with the system and as a consequence, they are able to predict the rest of the systems dialogue acts ahead of time.

To summarise, four TTP requiring incremental dialogue processing have been selected for rule-based implementation (one of them from both sides: system and user):

- FAIL_RAW (System side)

- INCOHERENCE_INTEPR (System side)

- FEEDBACK_RAW (System side)

- BARGE_IN_RESP (System and user side)

In Chapter 5, the details of the implementation, the rules chosen as well as a comparative study in a simulated environment are provided.
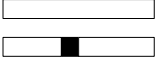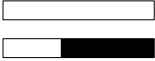
TTP	Phonetic act	Illocutionary act	Perlocutionary act	Motivations
FLOOR_TAKING_IMPL		I am about to start speaking	• Shift H's attention towards T	• Desire to start a conversation
DIALOGUE_INIT		I start this conversation and I inform you that x	• Shift H's attention towards T • Make H aware of the dialogue topic (x)	• Desire to start a conversation about x
FAIL_IMPL		I don't understand what you are talking about	• Make H stop • Make H repeat or reformulate	• Fix desynchro- nisation
FAIL_RAW		I don't understand what you are talking about	• Make H stop • Make H repeat or reformulate	• Fix desynchro- nisation
FAIL_INTERP		I don't understand what you are talking about because of x	• Make H stop • Make H repeat or reformulate • Making H aware of what is preventing T from understanding	• Fix desynchro- nisation • More efficiency by providing more precision about the problem
INCOHERENCE_IMPL		What you just said is problematic	• Make H stop • Make H reconsider what he just said	• Fix desynchro- nisation
INCOHERENCE_RAW		What you just said is problematic	• Make H stop • Make H reconsider what he just said	• Fix desynchro- nisation
INCOHERENCE_INTERP		What you just said is problematic because of x	• Make H stop • Make H reconsider what he just said • Making H aware of the problem in his utterance	• Fix desynchro- nisation • More efficiency by providing more precision about the problem
BACKCHANNEL		I understand (and sometimes: I agree)	• Make H continue	• More information from H
FEEDBACK_RAW		I understood that you said x	• Make H continue • Make H correct in case of a misunderstanding or not	• More information from H • Desire to confirm that H's utterance has been well understood

Table 2.3: Taxonomy labels (1/2)
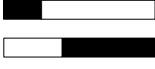
TTP	Phonetic act	Illocutionary act	Perlocutionary act	Motivations
FEEDBACK_INTERP		I understood that you said x that is related to y	• Make H continue • Make H correct in case of a misunderstanding or not	• More information from H • Desire to confirm that H's utterance has been well understood • Stronger confirmation by adding related information y
REF_IMPL		Yes, x	• Make H stop and understand T is referring to the last element he uttered (x)	• Selecting an option • Less effort as x has already been uttered by H
REF_RAW		Yes, x	• Make H stop and understand T is referring to the last element he uttered (x) • Make H correct in case of a misunderstanding	• Selecting an option • Less effort as x has already been uttered by H • Desire to confirm that the last option has been well understood
REF_INTERP		Yes, x that is related to y	• Make H stop and understand T is referring to the last element he uttered (x) • Make H correct in case of a misunderstanding	• Selecting an option • Less effort as x has already been uttered by H • Desire to confirm that the last option has been well understood • Stronger confirmation by adding related information y
BARGE_IN_RESP		x	• Make H aware of x	• Desire to move the dialogue forward • More efficiency because enough information has been provided before the end of the utterance
REKINDLE		The information you gave is not enough, please provide more information	• Make H resume	• Fix desynchronisation
END_POINT		x	• Make H aware of x	• Desire to move the dialogue forward

Table 2.4: Taxonomy labels (2/2)

Chapter 3

Turn-taking decision module: the Scheduler

3.1 Description

3.1.1 Overview

In the architecture introduced here, the five modules forming the dialogue chain (see Chapter 1) are split in two groups: those forming the *client* and those constituting the *service*. The ASR and the TTS are necessarily included in the client and the DM in the service. The NLU and the NLG can fit in both categories. This terminology has been borrowed to the computer network field where the client can refer to the user and to the application that interacts directly with the user in order to gather useful data for the interaction at the same time. Similarly, the server refers to the application that is in charge of handling user's requests, as well as the remote machine it is deployed on. In the case of dialogue systems, both parts can be embedded in the same device and they can also be distributed in two different machines.

Viewing traditional dialogue systems from this point of view translates into a ping-pong game, where the client sends a request which is processed by the service, and the latter sends a response back. The question tackled here is how to break this rigid mechanism in order to make the system able to process the user's speech incrementally. This chapter shows how, by starting from this new view of dialogue systems instead of the sequential one (dialogue chain), an incremental dialogue system can be derived from a traditional one at minimal cost. Moreover, in the resulting architecture, the turn-taking decision center is separated from the DM.

As illustrated in Fig. 3.1, a new interface is inserted between the client and the service (Khouzaimi et al., 2014). This new module is called the *Scheduler* (this denomination is borrowed from (Laroche, 2010)). It can be deployed on the same machine as the client, as the service or in a dedicated server. The objective is to make the set {Scheduler+service} behave like an incremental dialogue system from the clients point
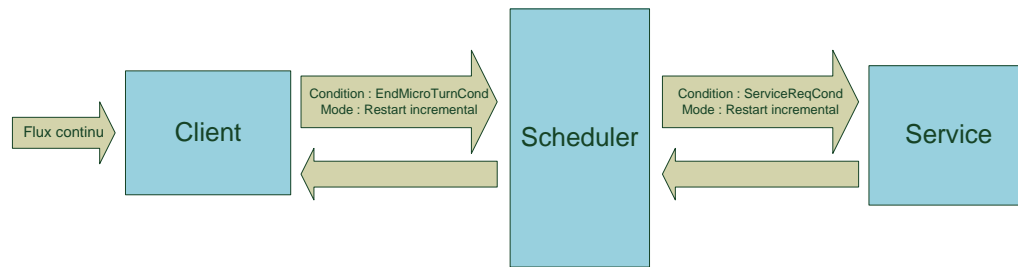
Figure 3.1: The Scheduler: an interface between the client and the service

of view, without modifying the initial functioning of the service. Therefore, a framework that can transform any dialogue system in its incremental version just by adding a new layer is provided.

This alternative way of designing incremental dialogue systems also has the advantage of clearly separating turn-taking management from the rest. As it will be seen along this thesis, turn-taking strategies will be implemented exclusively in the Scheduler (the rest of the system remaining the same). Our ultimate goal is to make this module learn optimal turn-taking behaviours by itself.

3.1.2 Time sharing

There are four types of human machine interfaces, given whether both communication channels (from the user to the system and the other way around) are punctual or discrete:

- **Discrete User/Discrete System:** This is the most basic way of human computer interaction. For example, the user hits a button and she immediately gets a response back. This is still widely used in many applications like browsers (when basic web browsing, without watching videos or listening to audio).

- **Discrete User/Continuous System:** Most of the currently deployed vocal platforms operate in this mode as they asked the user for DTMF inputs while they use natural language to provide instructions (using pre-recorded audio or speech synthesis).

- **Continuous User/Discrete System:** The application Shazam would be a good example of this communication mode. The user starts playing a song and the application listens. Once the latter recongnises it, it instantly displays the title and the singer on the screen.

- **Continuous User/Continuous System:** Spoken dialogue systems operate in a continuous/continuous mode. The communication signal from both sides is continuous.

For a system to be incremental, the user has to be continuous, therefore, the first and second communication modes are out of the scope of this thesis.
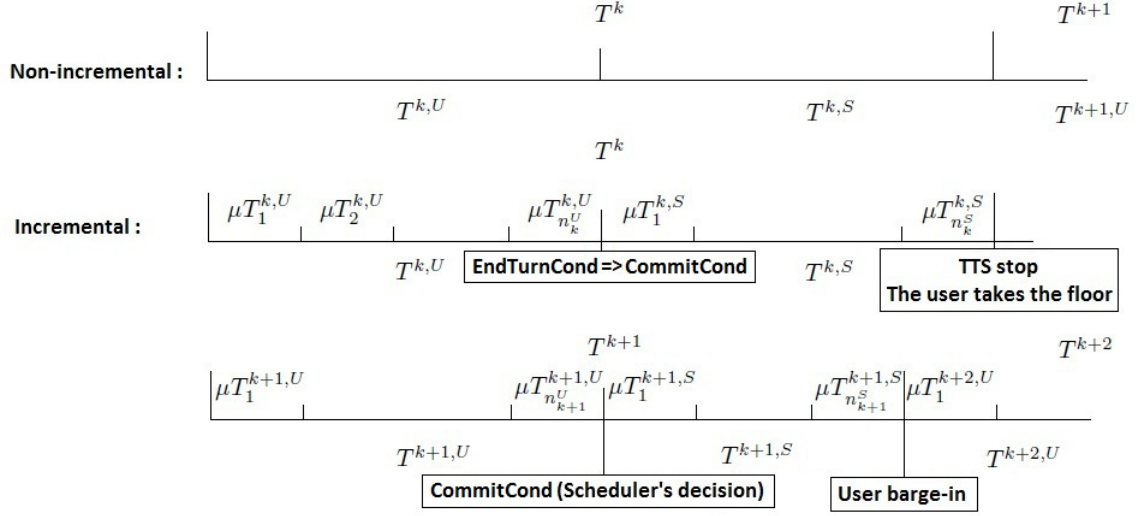
Figure 3.2: *Time sharing in traditional and incremental settings*

In traditional dialogue systems, time is shared in an ordered and clear manner. The dialogue is a simple sequence of turns $T^1, T^2...$, a turn being the time interval in which a user's utterance followed by the system's corresponding response takes place, or the opposite (depending whether the system adopts a user initiative or a system initiative strategy at each turn). For illustration and to simplify the notation, the system used here is supposed to belong to the first category, therefore, each turn is divided into two smaller time intervals, the user turn $T^{k,U}$ and the system turn $T^{k,S}$: $T^k = T^{k,U} \cup T^{k,S}$ (Fig. 3.2).

In this chapter, a few conditions are defined to precisely describe time allocation between the system and the user. The *activation time* of a condition refers to the exact moment when it goes from false to true. *EndTurnCond* is the condition that ends a user turn, it is generally assimilated to a long silence (Raux and Eskenazi, 2008; Wlodarczak and Wagner, 2013).

In incremental settings, this time sharing formalism does not hold anymore and a new condition should be defined: *EndMicroTurnCond* (with *EndTurnCond* $\Rightarrow$ *EndMicro-TurnCond*). The time interval separating two activation times of *EndMicroTurnCond* is called a *micro-turn*. As a consequence, the turn $T^{k,U}$ can be divided into $n^{k,U}$ micro-turns $\mu T_i^{k,U}$: $T^{k,U} = \bigcup_{i=1}^{n^{k,U}} \mu T_i^{k,U}$. The pth *sub-turn* of turn $T^{k,U}$ is defined as $T_p^{k,U} = \bigcup_{i=1}^{p} \mu T_i^{k,U}$.

The request that the user makes during $T^{k,U}$ is referred to as Req^k and the corresponding response is $Resp^k$. This architecture does not process incremental units like in (Schlangen and Skantze, 2011), instead, at each new micro-turn, it will take the whole information available since the beginning of the turn[1] (at the pth micro-turn, all what

---

[1]This way of manageing incremental dialogue is called *restart incremental* in (Schlangen and Skantze, 2011).

the user uttered during $T_p^{k,U}$). This *partial request* is called Req_p^k.

3.1.3 The Scheduler
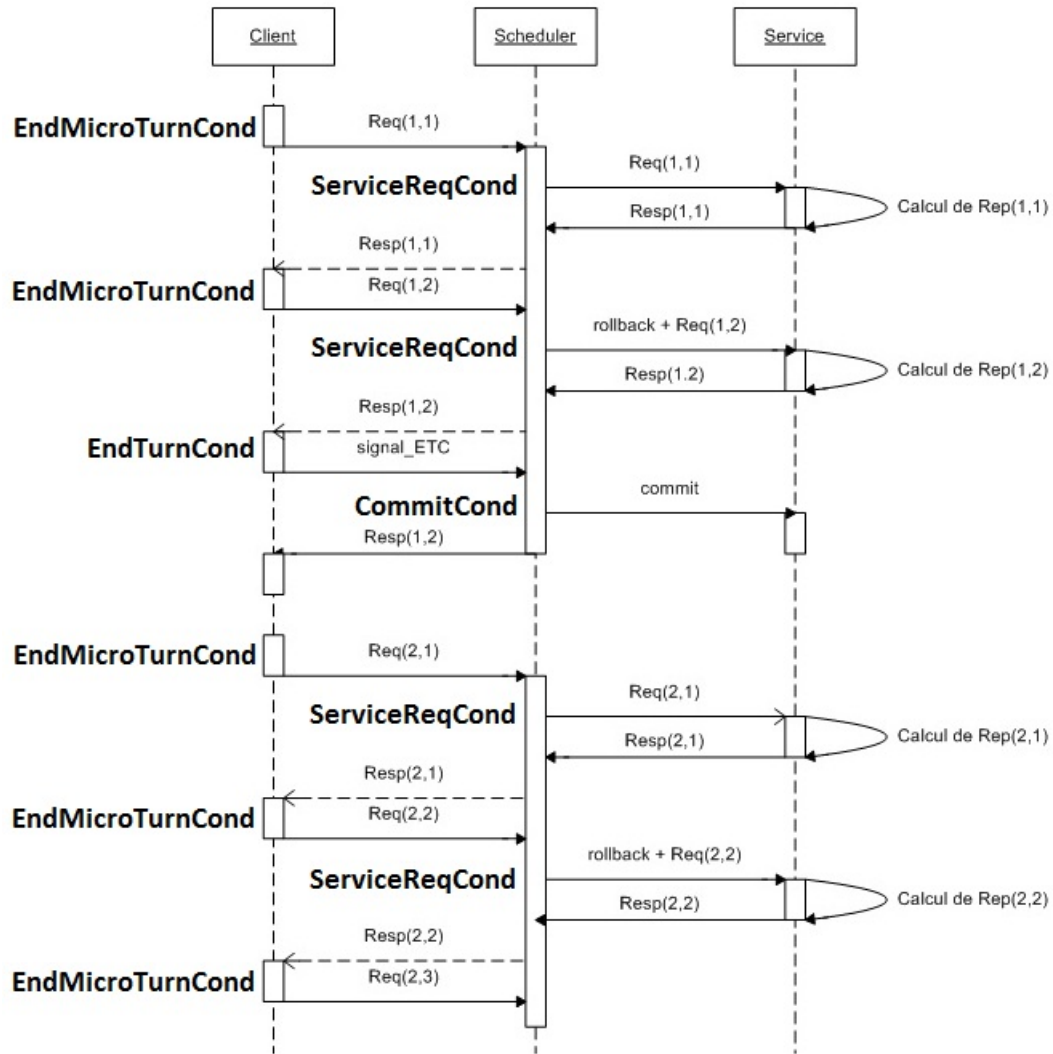
Figure 3.3: Incremental behaviour with the Scheduler

During the pth micro-turn of the kth user turn, the client sends Req_p^k to the Scheduler. The latter has to decide whether to send it to the service or not and the corresponding condition is called *ServiceReqCond*. A good example is *ServiceReqCond* = $(Req_p^k \neq Req_{p-1}^k)$ as sending the same request twice is useless. Then, the service provides the corresponding response $Resp_p^k$ and the Scheduler stores it. The key idea of this architecture is that the Scheduler decides whether to retrieve this response to the client (making it take the floor through the TTS) or not (waiting for more information to come from the client). This decision can also be forced by the client when sending an end of

turn signal signal_ETC, like a long enough silence for instance. The most interesting fact about the Scheduler is that it is able to decide when to take the floor without waiting for signal_ETC, the corresponding condition is called *CommitCond*. The Scheduler functioning over time is illustrated in Fig. 3.3.

Turn	User subturn	Input	Real context	Simulation context
T^1	$T_1^{1,U}$	Req_1^1	$\text{ctxt}(T^0)$	$\text{ctxt}(T^0 + T_1^{1,U})$
	$T_2^{1,U}$	Req_2^1	$\text{ctxt}(T^0)$	$\text{ctxt}(T^0 + T_2^{1,U})$
	...	...	$\text{ctxt}(T^0)$	...
	$T_{n^{1,U}}^{1,U}$	$Req_{n^{1,U}}^1$	$\text{ctxt}(T^0)$	$\text{ctxt}(T^0 + T_{n^{1,U}}^{1,U})$
	COMMIT: $ctxt(T^1) = ctxt(T^0 + T_{n^{1,U}}^{1,U})$			
T^2	$T_1^{2,U}$	Req_1^2	$\text{ctxt}(T^1)$	$\text{ctxt}(T^1 + T_1^{2,U})$
	...	...	$\text{ctxt}(T^1)$	...

Figure 3.4: Double context management: real and simulated

Nevertheless, this approach raises on technical problem. Most of the requests that are made to the service are only aimed to see what would be its response for certain partial utterances and they are discarded right after. However, they might modify the dialogue state in the service which is a side effect to be avoided. As a consequence, two dialogue contexts are maintained:

- **The real context:** The dialogue context as traditionally used in dialogue systems. Contains the data and the variables that are aimed to last and be used in the rest of the dialogue.

- **The simulated context:** A copy of the real context, at the pth micro-turn, $Resp_p^k$ could be useful for the dialogue or not. Therefore, only this context is modified at the first place, the Scheduler decides later whether to keep the changes in the real context or not.

These dialogue contexts are managed by two actions performed by the Scheduler:

- **Commit:** The Scheduler commits to a partial request and the corresponding response when it decides to deliver the latter to the client, hence taking the floor immediately and not waiting for any further information. In that case, the simulated context is saved into the real context (thus becoming the new reference).

- **Cancel or rollback:** The scheduler cancels the context changes when it decides to discard the last response obtained from the service. In that case, the real context is copied into the simulated one, rollbacking it to its original state. As shown in Fig 3.3, this decision is only made when a new - potentially more complete - partial request is received from the client.

The way the real and the simulated context are managed through the commit and the cancel actions is illustrated in Fig. 3.4.

3.2 Illustration

As a proof of concept, this section describes two instanciations of the previous abstract architecture, both in the case of a textual and a spoken dialogue system.

3.2.1 A textual dialogue system: CFAsT

Figure 3.5: *The incremental version of the CFAsT project. The traditional view is represented on the left and the new incremental one is depicted on the left.*

CFAsT stands for Content Finder AssistanT. This application, developed at Orange Labs, is aimed at automatically generating virutal assistants that help the user efficiently find specific content in databases (given as input). At each dialogue turn, the user provides some new information about his target and by using a keyword spotting algorithm, the system keeps narrowing the set of possibilites. The interface is made of a text box with a *validate* button. The dialogue service is deployed as a web service on a servlet container and the client is a javascript web page loaded on the user's browser.

In order to make it incremental, a Scheduler has been deployed as a servlet on the same container as the service and the javascript requests have been re-targeted towards it (the service is no longer directly visible by the client). Moreover, as discussed earlier, the dialogue context in the service have been duplicated. In this setup, *EndTurnCond*

corresponds to the event of clicking on *validate* (or hitting the carriage return). A word-based micro-turn configuration has been implemented, therefore, *EndMicroTurnCond* is the action of hitting the space bar. *ServiceReqCond* is not useful here, hence, it is true all the time.

This implementation is aimed to provide a first proof of concept of the methodology introduced here. However, signal_ETC = *CommitCond* = *EndTurnCond* (the Scheduler never takes commit decisions by itself). Therefore, this is a very limited illustration.

3.2.2 A spoken dialogue system: Dictanum

Number dictation is an activity that can be encountered in several kinds of applicative domains: phone numbers, credit cards...Analysing human conversations in such situations shows that the number at hand is often broken into small pieces (a few digits each) and a confirmation follows each one of them. Therefore, the number is incrementally grounded so that in the case of a misunderstanding, only the last dictated part has to be repeated. Traditional dialogue systems behave differently as they ask the user to enter the whole number at once. This translates into long and tedious dialogues as only one minor mistake is enough to make the user repeat the number from the beginning, often for several times. For example:

SYSTEM: Welcome to the Dictanum service. Please dictate your number.

USER: 01 45 65 79 98.

SYSTEM: The dictated number is 01 45 67 79 98. Is that correct?

USER: No.

SYSTEM: Sorry. Let us try again. Please dictate your number.

USER: 01 45 65 79 98.

SYSTEM: The dictated number is 01 47 65 79 98. Is that correct?

USER: No.

SYSTEM: Sorry. Let us try again. Please dictate your number.
    [...] *(given that the user is patient enough not to hang up)*

SYSTEM: The dictated number is 01 45 65 79 98. Is that correct?

USER: Yes.

SYSTEM: Thank you for using Dictanum. Bye.

With incremental dialogue processing, it is possible to design a strategy that is closer to the human behaviour. To show that, a number dictation dialogue system called Dictanum has been implemented using Disserto. It performs turn-based dialogue hence asking the user for the whole number at once. Here, the client is a web page displayed

on the browser that uses Google API for both the ASR and the TTS tasks and the service is a servlet deployed on a servlet container.

Afterwards, the approach introduced here has been used to build the incremental version of the system (like in the case of the CFAsT application, the Scheduler has been deployed as a servlet on the same container as the service). To do so, two silence duration thresholds have been defined: the short silence threshold δ_s and the long one Δ_s ($\delta_s \leq \Delta_s$). *EndMicroTurnCond* is triggered when a short silence is detected and similarly, *EndTurnCond* corresponds to long silences. A feedback is delivered to the user in the case of a short silence (repeating the last 4 digits). If the user ignores the feedback and keeps dictating his number, the system keeps on adding digits to its list, however, if the user starts his next utterance with *No*, the feedback content is deleted from the number. Here is a dialogue example:

SYSTEM: Welcome to the Dictanum service. Please dictate your number.

USER: 01 45 *(short silence)*

SYSTEM: 01 45

USER: 65 79 *(short silence)*

SYSTEM: 67 79

USER: No, 65 79 *(long silence)*

SYSTEM: Sorry, 65 79

USER: 98 *(short silence)*

SYSTEM: 98

USER: ... *(long silence)*

SYSTEM: The dictated number is 01 45 65 79 98. Is that correct?

USER: Yes.

SYSTEM: Thank you for using Dictanum. Bye.

It is interesing to notice that if $\delta_s = \Delta_s$, the user is likely to dictate his number in one shot, never waiting for a feedback. Therefore, moving δ_s between 0 and Δ_s creates a continuum between the traditional and the incremental version.

Dictanum also offers the possibility for the user to interrupt the system during the final feedback, in order to make local corrections. To do that, this feedback is sent to the TTS in the following format: *The dictated number is 01 <sep> 45 <sep> 65 <sep> 79 <sep> 98. Is that correct?*. The latter pronounces the sentence chunk after chunk (chunks are delimited using the separator <sep>), each chunk lasting for the same number of micro-turns. This leads to the following kind of strategy:

SYSTEM: The dictated number is: 01 45 67...

USER: No, 65.

SYSTEM: Sorry. The dictated number is 01 45 65 79 98. Is that right?

USER: Yes.

SYSTEM: Thank you for using Dictanum. Bye.

3.3 Discussion

3.3.1 Levels of incrementality

Dialogue systems can be classified in four categories given the way they integrate incremental behaviour. The first category is made of traditional systems (Laroche et al., 2011). Then comes the second category where traditional systems locally simulate a few incremental behaviours. For instance, in (El Asri et al., 2014b), the system enumerates a list of options and the user selects the one that fits him best by uttering *Yes* or *Ok* for example (REF_RAW in the taxonomy introduced in Chap. 2). The architecture introduced in this thesis belongs to the third category where incremental behaviour is obtained based on modules that are innately non-incremental (the service in our case). Other examples are described in (Selfridge et al., 2012) and (Hastie et al., 2013). Finally, the fourth category is made of incremental dialogue systems that are constituted of fully-incremental modules. In (Schlangen and Skantze, 2011), an abstract model for incremental architectures is presented where all the categories can fit, but the work that has been pursued by the authors and their research groups later on goes along with the spirit of this last category.

Categories 2, 3 and 4 embed different features related to incremental behaviour as summarised in Fig. 3.1.

3.3.2 Enhancing a traditional dialogue system's turn-taking abilities at a low cost

Adopting the sequential paradigm described in Chapter 1 is a natural way of designing incremental dialogue systems (Schlangen and Skantze, 2011). The dialogue chain is kept unchanged, however, an important amount of work has to be done in order to design an incremental version of each one of the modules. The approach introduced in this chapter makes it possible to build an incremental dialogue system starting from a traditional one instead of starting from scratch. Therefore, the development cost is significantly reduced and moreover, the resulting incremental dialogue system obtained benefits from all the experience and the adjustments embedded in the original dialogue system. In this section, an analysis is led to determine which challenges brought by building incremental dialogue systems are simplified when using the approached introduced in this chapter.

Incremental ASR is a prerequisit for the implementation of a Scheduler-based architecture. Therefore, it is not simplified by this approach and an inaccurate, slow or

unstable ASR module still hurts the dialogue quality in the same way. In terms of NLU, two cases have to be distinguished: putting the SCheduler before the NLU or after it. In the first case, it does not make sense to use an incremental NLU as the Scheduler proceeds on a restart incremental fashion (sending the whole user's partial utterance at each micro-turn). However, in the second case, it is possible to benefit from the advantages of incremental NLU (forming concepts in a real incremental way resulting in more efficient processing). In that case, the Scheduler receives a sequence of potentially unstable set of concepts.

The core difference between both approaches resides in the DM task. In a full-incremental architecture, dialogue act and turn-taking decisions and intertwined. The DM receives the input concepts chunk by chunk, and at each micro-turn, the new information can be viewed as the continuity of what has been understood so far, or as a signal pushing the DM to revoke the current hypothesis before taking a new action. In the Scheduler-based approach, as the restart incremental paradigm is adopted, the revoke mecanism is intrinsicly implemented as it is performed beforehand (the ASR changing its best hypothesis is a case of revoke).

Incremental NLG does not make sense in the frame proposed here as the Scheduler can only retrieve full dialogue acts that have been precomputed by the service. As a consequence, the system can change its mind about the utterance currently uttered but it can only generate a new dialogue act from scratch with is potentially not related to the previous one. Nevertheless, as discussed in Chapter 1, this is not a major problem when it comes to spoken dialogue systems relying on voice as the only interaction modality. It is mostly used when the system produces comments about a moving object or situation it is observing (Baumann and Schlangen, 2013).

Finally, as far as the TTS task is concerned, the same remark as for the NLG can be made. However, it can be interesting to give the Scheduler the power to stop the TTS. This is particularly interesting when the DM decides to revoke its current belief.

Features	Category 1	Category 2	Category 3	Category 4
TTS interruption after input analysis	-	+	+	+
Link interruption time with TTS	-	+	+	+
User interruption by the system	-	-	+	+
Better reactivity	-	-	+	+
Optimal processing cost	-	-	-	+

Table 3.1: Available features for dialogue systems given the way they integrate incrementality

3.3.3 Separating dialogue management from floor management

Chapter 4

Incremental dialogue simulation

4.1 Agenda management task

A personal agenda assistant has been implemented as our task for the experiments. The user can add, move or delete events in his agenda. For instance, a request could be: *I would like to add the event football game on March 3rd from 9 to 10 pm*[1]. This is a slot filling task with four slots:

- **ACTION:** The type of action the user wants to perform. Can take three different values: ADD, MODIFY or DELETE.

- **DESCRIPTION:** The title of the event.

- **DATE:** The date of the event.

- **SLOT:** The time slot of the event.

  However, the no overlap is tolerated between events in the agenda.

  The US is given two lists of events: *InitList* and *ToAddList*. The first one contains the events that already exist in the dialogue before the dialogue and the second one contains the ones that the US is supposed to add during the dialogue. Each event is associated with a priority value and the US must prefer adding the ones with high probability first. Its aim is to make as many events as possible fit in the agenda.

4.2 The service

4.2.1 Natural Language Understanding

A recursive algorithm transforms the user's utterance hypothesis into a concept tree. To do that, a set of rules have been defined. Each rule transforms a word, a concept or

---

[1]The dialogues are actually in french but they are translated in English in this thesis

Table 4.1: *NLU rules types*

Rule type	Description	Example
Tag	Words are associated to a label	remove : [DELETE]
Regular expressions	Spots words that satisfy a regular expression	Regex([0-9]+) : NUMBER($word)
Combine	Words and concepts are mapped into a new concept	Combine(NUMBER,MONTH) : DATE

any combination of the two into a new concept. Three types of rules are used; they are depicted in table 4.1

The NLU algorithm keeps applying these rules the order they appear in the rules file until two consecutive results are identical. For instance, parsing the sentence *I want to add the event birthday party on January 6th from 9pm to 11pm* is performed following these steps:

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(janvier|...|decembre) : MONTH($word)

   - Regex([0-9]+) : NUMBER($word)

   - Regex((([0-1]?[0-9])|(2[0-3]))h([0-5][0-9])?) : TIME($word)

2. **I want to ADD the TAG_EVENT birthday party on DATE(6,1) SLOT(TIME(21,0),TIME(23,0))**

   - Combine(NUMBER,MONTH) : DATE(NUMBER,MONTH)

   - Combine(from,TIME_1,to,TIME_2) : SLOT(TIME_1,TIME_2)

3. **I want to ADD EVENT(birthday party, DATE(6,1), SLOT(TIME(21,0),TIME(23,0)))**

   - Combine(TAG_EVENT,$x,on,DATE,SLOT) : EVENT($x,DATE,SLOT)

4. **I want ACTION(ADD, EVENT(birthday party, DATE(6,1), SLOT(TIME(21,0),TIME(23,0))))**

   - Combine(ADD,EVENT) : ACTION(ADD,EVENT)

4.3   User simulator

4.3.1   Overview

The User Simulator (US) is composed of six modules (see Figure 4.1). The Intent Manager, the NLU, the Verbosity Manager, the ASR output simulator, the NLG and the

Patience Manager. In every incremental dialogue setup, each dialogue turn is divided into smaller micro-turn given a particular criteria; a micro-turn could be a time window (500 milliseconds for example), it could also be word-based or concept-based (a new word or a new concept marks the end of a micro-turn). The US described here is word-based.

At each micro-turn, the US generates a partial N-Best. It embeds the whole utterance from the beginning of the turn (*restart incremental* mode (Schlangen and Skantze, 2011)). On the other hand, either the US receives an answer from the Scheduler at a certain micro-turn and it stops speaking, either it does not and it continues speaking if it has additional thing to say, or spontaneously releases the floor. When the dialogue lasts for too long without achieving the task at hand, the US can end the dialogue.

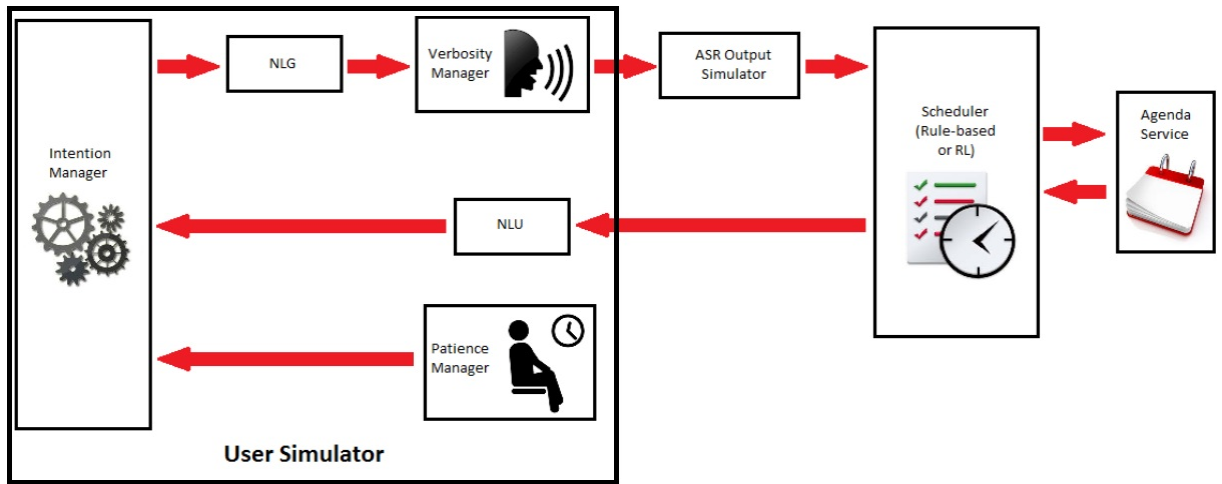In the following, the different components of the US are described in detail.

Figure 4.1: Simulated environment architecture

4.3.2 Intent Manager

The aim of the US is to make the biggest number possible of events taken from *InitList* and *ToAddList* fit in the agenda. The events with the highest priority are considered first. To do so, the Intent Manager runs Algorithm 1 where

- **actionList:** the list of actions to be performed by the system (initially the list of ADD actions corresponding to *ToAddList*).

- **act():** function that performs a list of actions.

- **perform():** function that performs a single action.

- **alternatives():** function that returns the alternative events of the one corresponding to the input action.

- **conflictsByAlt:** contains couples (event,conflictSet) mapping events to the sets of conflicting events in the agenda.

- **maxPrioByAlt:** contains couples (event, maxPrio) mapping events to the maximum priority in conflictsByAlt(event).

4.3.3   NLG and Verbosity Manager

The NLG module transforms action concepts as well as date, slot, description and yes/no concepts into simple straight forward utterances. For instance the NLG result for the intent ACTION(ADD, EVENT(birthday party, DATE(6,1), SLOT(TIME(21,0),TIME(23,0)))) is *add the event birthday party on January 6th from 9pm to 11pm*.

Nevertheless, compared to human/human conversations, limiting interactions to this kind of simple utterances is not realistic. Therefore, they are enhanced in the Verbosity Manager with prefixes like *I would like to*, *Is it possible to*...and suffixes like *if possible*, *please*... In (Ghigi et al., 2014), a corpus study showed that users tend to go off-domain and to repeat the same information several times in the same sentence. These behaviours are also replicated in the Verbosity Manager: 10% of the times, the US utters an off-domain sentence, and 30% of the times where there is a misunderstanding, it repeats the same information twice. These proportions have been estimated based on the corpus study mentioned before.

4.3.4   ASR output simulator

The ASR output simulator generates an N-Best that is updated at each new micro-turn. For instance, if at a certain point, the US uttered *I would like to add the event birthday party on...*, a possible N-Best could be (the numbers between brackets represent ASR scores):

- (0.82) I would like to add the event birthday party on

- (0.65) I like to add the event birthday party on

- (0.43) I have had the event birthday party

- (0.33) I would like to add the holiday party

- (0.31) I like to add the holiday party on

More formally, at time t, the N-Best is an N-uplet $(s_1^{(t)}, hyp_1^{(t)}), ..., (s_N^{(t)}, hyp_N^{(t)})$. At time t+1, a new word w_{t+1} is send to the ASR output simulator and the latter calculates a new associated N-Best. Therefore, at this stage, the system has two N-Best (one for the whole current partial utterance and one for the new word) that it has to combine to obtain the new partial utterance N-Best. In the following how the word N-Best is calculated and how it is incorporated in the partial utterance N-Best.

Algorithm `act` (*actionList*)

 if *actionList not empty* **then**
  added ← perform(actionList(0))
  if *not added* **then**
   alt ← alternatives(actionList(0))
   conflictsByAlt ← []
   maxPrioByAlt ← []
   addedAlt ← false
   while *alt not empty AND not addedAlt* **do**
    addedAlt ← perform(alt(0))
    if *not addedAlt* **then**
     conflictsByAlt ← [conflictsByAlt (alt(0),conflicts(alt(0)))]
     maxPrioByAlt ← [maxPrioByAlt (alt(0),max priority of conflicts(alt(0)))]
     alt.remove(0)
    end
   end
   if *not addedAlt* **then**
    selectedAlt ← argmin maxPrioByAlt
    movedToAlt ← true
    while *conflictsByAlt no empty AND movedToAlt* **do**
     moveActionList ← list of move actions to the different alternatives of conflict
     movedToAlt ← act(moveActionList)
    end
    if *movedToAlt* **then**
     perform(ationList(0))
    else
     return *false*
    end
   end
  end
  actionList.remove(0)
  return *act(actionList)*
 end
 return *true*

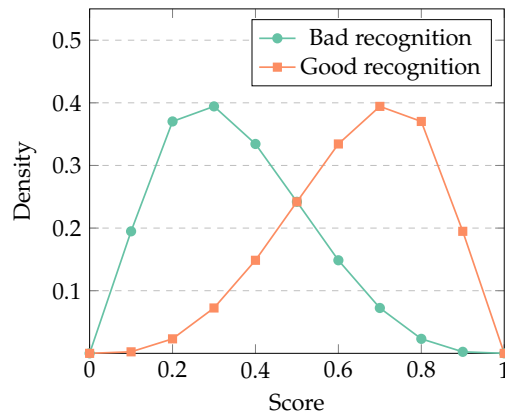**Algorithm 1:** Intent Manager algorithm

Figure 4.2: *ASR score sampling distribution*

A Word Error Rate (WER) is given as a parameter to the ASR output simulator. It controls the noise level that one wants to simulate. A Scrambler is used to generate a modified version of some selected words. It can achieve three types of modifications:

- Replace the word with a different word taken randomly from a dictionary (probability: 0.7).

- Add a new word (probability : 0.15).

- Delete the word (probability : 0.15)

Starting from here, we describe the algorithm used to generate the N-Best associated with a single word.

1. Determine whether w_{t+1} is among the N-Best or not with a probability that is computed as follows: (1 - WER) + INBF.WER, where INBF (In N-Best Factor) is a parameter between 0 and 1 (set to 0.7 in this thesis). If w_{t+1} is not in the N-Best, then the latter contains only scrambled versions of this word and we directly jump to step 4.

2. The first hypothesis is set to be w_{t+1} with a probability of (1-WER), otherwise, it is a scrambled version of it.

3. If the the first hypothesis is not w_{t+1}, then this word's position is randomly chosen between 2 and N. Moreover, the other hypothesis are scrambled versions of it.

4. The confidence score associated to the best hypothesis (s_0) is sampled as sigmoid(X) where X is a gaussian. The distribution mean is -1 if the first hypothesis is wrong and 1 when it is right. The standard deviation of X is one. By taking the sigmoid, this leads to two distributions (depicted in Figure 4.2) with a mean of 0.3 and 0.7 and a standard deviation of 0.18 for both (which can be changed to simulate different levels of accuracy of the confidence score model. The closest to 0, the better the model).

5. The scores for the other hypotheses are computed in an iterative way. For *i* be-

tween 2 and N, s_i is uniformly sampled in $[0, s_{i-1}]$.

In incremental settings, the ASR input is a continuous audio stream. In reality, it is split into small consecutive chunks of data. At each new input arrival, the output is updated. However, a new chunk of audio data in the input does not necessarily translate into a new chunk of text added to the output. In reality, the latter can be partially or even completely changed while update. An example from (Schlangen and Skantze, 2011) is when the user utters the word *forty*: the system first understands *four* and then *forty*. The transition between the two hypotheses does not translate into an addition of information but the first hypothesis has to be revoked and replaced by the second one. This phenomenon is referred to as the *ASR instability*.

To replicate this behaviour, a language model is needed to compute the scores corresponding to the different hypotheses in the N-Best. Therefore, for sentences that are more in alignment with the model will have high scores thus pushed to the top of this N-Best. Here, we use the NLU knowledge as a proxy for the language model by making the following assumption: *the more an utterance generates key concepts once fed to the NLU, the more it is likely to be the correct one*. Therefore, as soon as a new action, date or time slot concept is detected in hyp_i, s_i is boosted as follows:

$$s_i \leftarrow s_i + BF.(1 - s_i)$$

where BF is the Boost Factor parameter. Here it is set to 0.2. An illustration of this mechanism is provided in Fig. 4.3.

4.3.5 Timing and patience manager

When it comes to incremental processing, timing is key. However, the main objective of simulation is to generate dialogues in an accelerated mode, hence, no track of timing is kept. In order to approximate durations, the user's and the system's speech rates are considered to be 200 words per minute (Yuan et al., 2006).

Users tend to get impatient, at various degrees, when dialogue systems take too long to accomplish the task they are asked for. To simulate this behaviour, a duration threshold is chosen at each new dialogue task (adding, modifying of deleting an event) that will cause the user to hangup when reached. It is computed as follows

$$d_{pat} = 2\mu_{pat}.sigmoid(X) \tag{4.1}$$

where X follows a gaussian distribution of mean 0 and variance 1 and consequently μ_{pat} is the mean duration threshold.

Current sentence uttered : *I want to add the event birthday party on January*
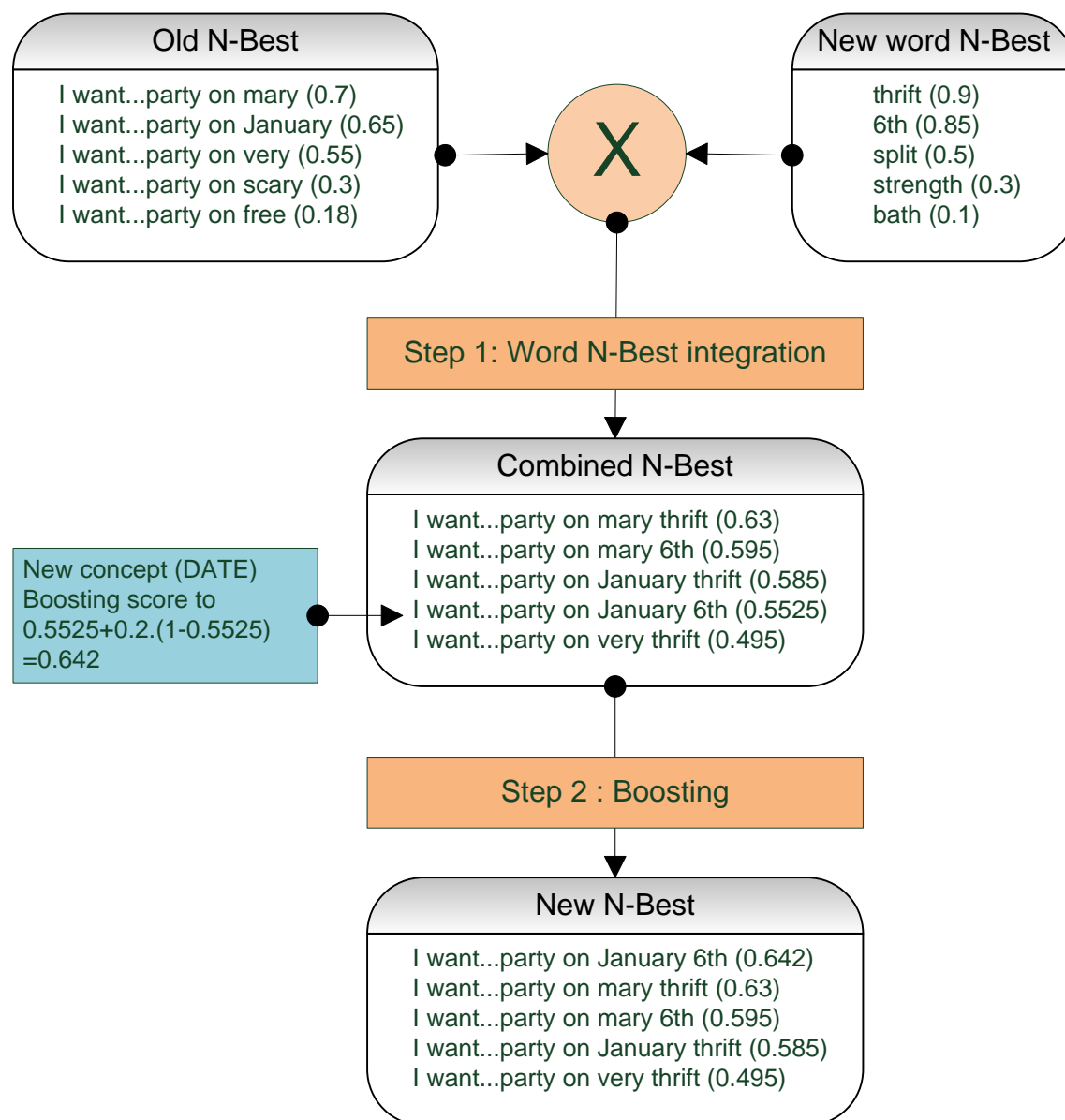**New word added :** *6th*

Old N-Best

I want...party on mary (0.7)
I want...party on January (0.65)
I want...party on very (0.55)
I want...party on scary (0.3)
I want...party on free (0.18)

X

New word N-Best

thrift (0.9)
6th (0.85)
split (0.5)
strength (0.3)
bath (0.1)

Step 1: Word N-Best integration

Combined N-Best

I want...party on mary thrift (0.63)
I want...party on mary 6th (0.595)
I want...party on January thrift (0.585)
I want...party on January 6th (0.5525)
I want...party on very thrift (0.495)

New concept (DATE)
Boosting score to
0.5525+0.2.(1-0.5525)
=0.642

Step 2 : Boosting

New N-Best

I want...party on January 6th (0.642)
I want...party on mary thrift (0.63)
I want...party on mary 6th (0.595)
I want...party on January thrift (0.585)
I want...party on very thrift (0.495)

Figure 4.3: *An illustration of the incremental ASR output N-Best update*

Chapter 5

Handcrafted strategies for improving dialogue efficiency

5.1 User and system initiative

5.1.1 Strategies

Most of the current task oriented dialogue systems operate in a slot-filling manner. Here, we identify three strategies to fill all the slots needed by the system:

- **System Initiative (SysIni):** The system asks the user for the different slot values one by one, one at each dialogue turn like in the following example:

  SYSTEM: What kind of action do you want to perform?

  USER: Add.

  SYSTEM: Please specify a title.

  USER: Dentist.

  SYSTEM: Please specify a date.

  USER: March <noise>.

  SYSTEM: Sorry I don't understand.

  USER: March 7th.

  SYSTEM: Please specify a time slot.

  USER: From 10am to 11am.

  SYSTEM: Ok. So you want to add the event dentist on March 7th from 10am to 11am. Is that right?

  USER: Yes.

This strategy is very tedious, however, it has the advantage of baby-stepping the dialogue and recovering the errors progressively.

- **User Initiative (UsrIni):** The user is asked by the system to make a full sentence with all the information needed to process the request. For example:

SYSTEM: What can I do for you?

USER: I would like to add the event dentist on March <noise> from 10am to 11am.

SYSTEM: Sorry I don't understand.

USER: I would like to add the event dentist on March 7th from <noise> to 11am.

SYSTEM: Sorry I don't understand.

USER: I would like to add the event dentist on March 7th from 10am to 11am.

SYSTEM: Ok. So you want to add the event dentist on March 7th from 10am to 11am. Is that right?

USER: Yes.

With this strategy, the dialogue can progress much faster, nevertheless, unlike SysIni, when there is still missing information (even one slot), the user has to repeat all his request.

- **Mixed Initiative (MixIni):** Just like UsrIni, the system asks for all the slots at once but if there is still missing information, it switches to SysIni in order to gather the missing slot values one by one. For example:

SYSTEM: What can I do for you?

USER: I would like to add the event dentist on March <noise> from 10am to 11am.

SYSTEM: Please specify a date.

USER: <noise>.

SYSTEM: Sorry I don't understand.

USER: March 7th.

SYSTEM: Ok. So you want to add the event dentist on March 7th from 10am to 11am. Is that right?

USER: Yes.

By elaborating such a hybrid strategy, we hope that it will ally both advantages of SysIni and UsrIni. In the following, we use the simulator to investigate the performance of these strategies under different conditions of noise.
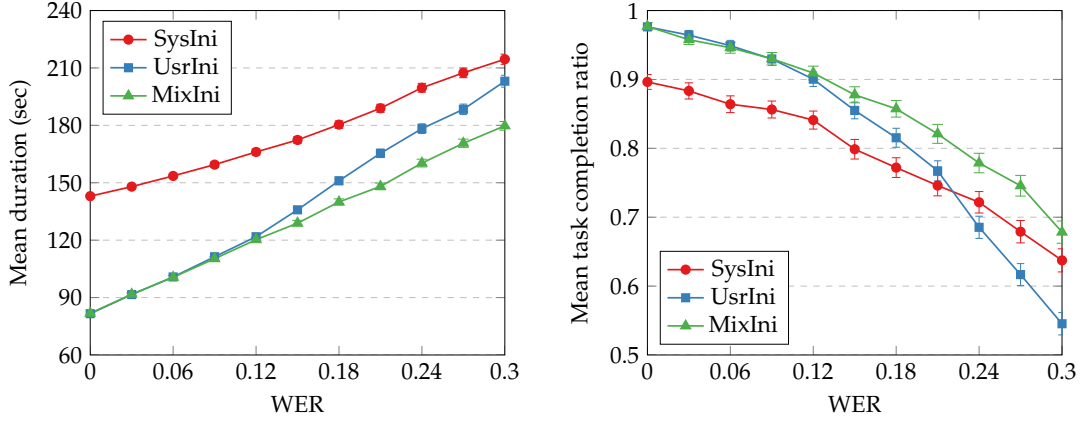
Figure 5.1: *Simulated mean duration (left) and dialogue task completion (right) for different noise levels*

Table 5.1: *The three scenarios used in the simulation*

Scenario	Title (Priority)	Date	Slot	DateAlt1	SlotAlt1	DateAlt2	SlotAlt2
1-Init	Guitar lesson(4)	November 17th	14:00-15:30	November 15th	9:45-11:15		
1-ToAdd	Book reading(8)	November 19th	10:30-12:30	November 14th	9:30-11:30	November 18th	16:30-18:30
	Watch the lord of the rings(12)	November 13th	9:30-12:30	November 15th	11:15-14:15		
2-Init	Guitar lesson(4)	November 17th	14:00-15:30	November 15th	9:45-11:15		
2-ToAdd	Tennis(5)	November 17th	13:15-15:15	November 19th	15:15-17:15		
	Gardening(9)	November 18th	13:15-15:15	November 14th	12:30-14:30		
3-Init	Guitar lesson(4)	November 17th	14:00-15:30	November 15th	9:45-11:15		
	Holidays preparation(1)	November 16th	12:30-14:30	November 17th	12:15-14:15		
	House cleaning(6)	November 13th	14:15-16:15	November 17th	15:30-17:30		
3-ToAdd	Give back book(7)	November 16th	14:00-14:30	November 13th	14:00-14:30		

5.1.2 Experiments

For the experiments, three dialogue scenarios have been used. As described in 4.3.2, a scenario is specified by two lists of events: *InitList* and *ToAddList*. The lists corresponding to the three scenarios are given in Table 5.1.

We vary the WER between 0 and 0.3 with a step of 0.03 in order to analyse the effect of noise on the different strategies. For each noise level, the three scenarios and run 1000 times. The mean duration of the dialogues, their task completion as well as the corresponding 95% confidence intervals are depicted in Fig. 5.1.

In low noise setups, SysIni is clearly less efficient as UsrIni; the dialogues take twice more time to finish with a lower completion rate (actually, it is a consequence of longer dialogue durations given the way the Patience Manager is designed). Nevertheless when the noise level reaches about 0.2, SysIni gives offers better completion rates. The duration is still lower in spite of the correlation between the two metrics. This is due to the fact that the durations distribution for UsrIni is centered on short dialogues whereas the distribution for SysIni is centered on average ones. Finally, MixIni seems to be the best strategy as it allies both the advantages of UsrIni and SysIni.

5.2 Incremental strategies

In this study, the two metrics under focus are duration and task completion (used as a proxy to measure dialogue efficiency). Nevertheless, one of the main motivations behind incremental dialogue process is to increase human likeness and the dialogue fluency by achieving smooth turn-taking. Some TTPs from the taxonomy established in Chapter 2 are more likely to improve these last aspects instead of the metrics that interest us most directly, like the BACKCHANNEL for instance. Therefore, only a few TTPs from the taxonomy established in Chapter 2 have been implemented: those that are more likely to improve our two objective metrics. However, the phenomena that are not studied here could have an indirect impact on duration and task completion as objective and subjective metrics are somehow correlated. In simulation, this is not visible though.

INIT_DIALOGUE and END_POINT are two phenomena that already exist in traditional dialogue systems. In the Scheduler module (see Chapter 3), we add the following phenomena: FAIL_RAW, INCOHERENCE_INTERP, FEEDBACK, BARGE_IN_RESP. The last phenomena have also been implemented in the US to make it able to interrupt the system. Section 5.2.2 provides the implementation details of these TTPs.

5.2.1 ASR instability

Because of the ASR instability phenomenon explained in 4.3.4, the current partial utterance is not guaranteed to be a prefix of future partial utterances and it is likely to change. However, the words at the end of this partial utterance are more likely to change than the ones at the start. In (McGraw and Gruenstein, 2012), it is shown that words that lasted for more than 0.6 seconds have 90% chance of staying unchanged. In this work, the speech rate (user and system) is supposed to be 200 words per seconds (Yuan et al., 2006) so 0.6 seconds corresponds to two words. Based on that, the handcrafted TTP implementation takes decisions by looking at the current partial hypothesis without its last two words (called the *the last stable utterance*). This margin is called the *Stability Margin* (SM = 2).

5.2.2 TTP implementation

In the following, the TTP implementation is described. The settings and the parameter values are handcrafted given our experience. In the following chapters, we will try to optimise them from data.

FAIL_RAW: Happens when the user speaks for two long with no key concept detected in her speech. It depends on the system's last question (the type of information it is waiting for). Here, it is triggered when the system asks an open question (all the slots at once) and the user's utterance contains 6 words with no action type detected

(add, modify or delete). This threshold is set to 3 in the case of yes/no questions, 4 for date questions and 6 for slot questions.

INCOHERENCE_INTERP: The system can fully understand the user's partial utterance and yet have a good reason to barge-in. This is the case when the user injects new information that is not coherent with the dialogue context. For instance, if the user says *I want to move the event football game to January...* and there is no event football game in the agenda, then letting him finish his utterance is a waste of time. As a consequence, it makes sense for the system to barge-in. In this implementation, this event is triggered as soon as the last stable utterance generates an overlap with an existing event in the agenda, or it tries to move or delete a non existing event.

FEEDBACK: This phenomenon translates into repeating a part if not the whole user's utterance. However, for the sake of simplicity, we limit ourselves here to the case where only the last word of the last stable utterance is repeated. The moments when feedbacks are performed in this implementation are the moments when the ASR confidence score drops below a certain threshold. To isolate the confidence score corresponding to the target word only, the ratio s_{t-SM}/s_{t-SM-1} is computed (as the confidence score of a sentence is the product of the scores associated with the words in this sentence). Here, this event is triggered whenever this ratio drops under 0.7.

BARGE_IN_RESP (System): Depending on the last system dialogue act (apart from dialogue acts reporting errors), the system can choose to barge-in once it has all the information needed to provide an answer. Again, it should also wait for the SM.

BARGE_IN_RESP (User): When the user gets familiar with the system, it tends to predict the system's dialogue act before the system finishes its sentence. Unlike the previous phenomena, this one is due the a user's decision. Hence, it has been implemented in the US.

5.2.3 Experiments

In this experiment, the TTPs phenomena described in Sect. 5.2.2 are used on the top of the slot-filling strategies introduced in Sect. 5.1.1. Incremental processing is useful when the user makes long utterances. This is not the case in the SysIni strategy where her utterances are very short. Therefore, incremental behaviour have only been added to UsrIni and MixIni to form two new strategies: UsrIni+Incr and MixIni+Incr. The associated performances are depicted in Fig. 5.2.

Adding mixed initiative behaviour or incrementality to UsrIni are both ways to improve its robustness to errors. Fig. 5.2 shows that in our case, incrementality is more

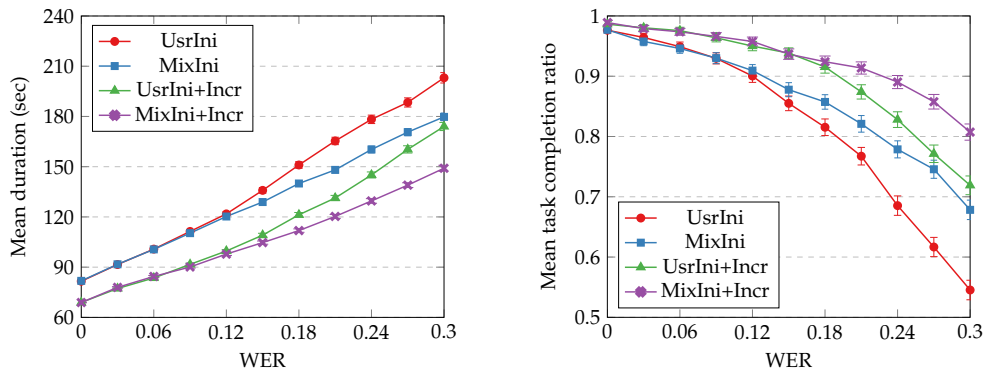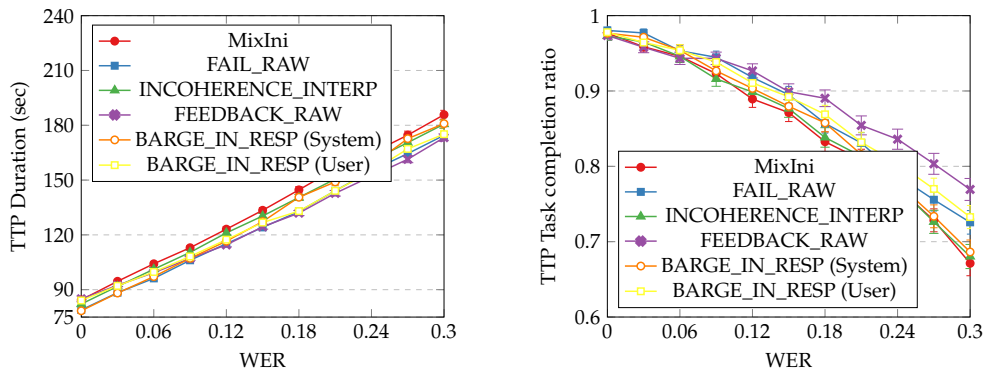*Figure 5.2: Mean dialogue duration and task completion for generic strategies.*

Figure 5.3: Mean dialogue duration and task completion for different turn-taking phenomena.

efficient. Most importantly, it is shown here that MixIni and incremental behaviour can be combined to form the best strategy.

As already mentioned in Chapter 2, the main objective of our TTP taxonomy is to break human dialogue turn-taking into small pieces, and hopefully get a better understanding of it. To illustrate this approach, we take a deeper look at MixIni+Incr by isolating its different components: FAIL_RAW, INCOHERENCE_INTERP, FEEDBACK, BARGE_IN_RESP (User) and BARGE_IN_RESP (System). The results reported in Fig. 5.3 show that FEEDBACK contributes the most to improve the baseline followed by BARGE_IN_RESP (User) and FAIL_RAW. INCOHERENCE_INTERP and BARGE_IN_RESP (System) seem to have no effect. This is due to the fact that in general, to detect an incoherence, one must wait until the end of the utterance (same requirement for detecting all the information needed to barge-in and provide an answer). One might argue that in some case, the US tries to refer to a non existing event (in the case of a MODIFY or DELETE action), therefore triggering an incoherence. In reality, our service is able to recognise an existing event even if only a prefix of its title is recognised. As a consequence, INCOHERENCE_INTERP is rarely triggered in the middle of a request. In order to force that case to occur more often, a scenario where the US has to try to move an event five times before encountering a free slot has been designed. The results are shown in Fig. 5.4: this time, INCOHERENCE_INTERP has a true added value.

Figure 5.4: *INCOHERENCE_INTERP evaluated in a more adapted task*
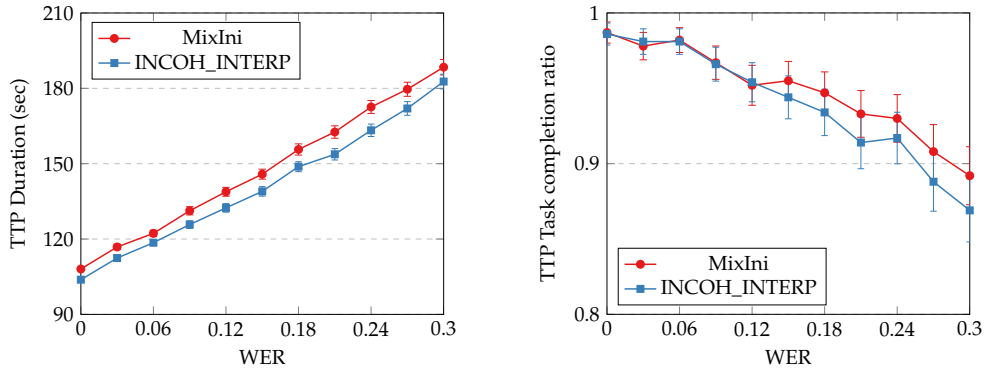
This little experiment raises an important point when it comes to studying efficiency in task oriented dialogues and as far as turn-taking mechanisms are concerned. The nature of dialogue is very diverse, therefore, results and the following conclusions (whatever the chosen metric is) should not be given separately from the dialogue frame. Here, the focus is efficiency, however, human-machine dialogue can also be used in a non task oriented fashion. The metrics chosen here make no longer pertinent in such a frame and it would make more sense to consider subjective ones. Moreover, even in task oriented dialogue, booking a train is very different from dictating a text message. In the first case, a keyword spotting NLU is adapted to the task at hand but in the second situation, the system does try to understand what the user is saying. The feedback can be very useful in this situation for instance.

Chapter 6

Reinforcement learning for turn-taking optimisation

6.1 Model

6.1.1 State representation

The following features are used to describe the system state:

- **SYSTEM_REQ:** At each dialogue turn, the system is requiring a particular information. For instance, after an open question, it is waiting for all the slot values to be provided at once but it can also be waiting for a specific question or a response to a confirmation. This feature refers to the information that it is waiting for at this moment. It can take 6 different values.

- **LAST_INCR_RESP:** As described in Chap. 3, the Scheduler stores the last response it gets from the service at each micro-turn. We use that as our second feature. 11 different values can be taken by this feature.

- **NB_USER_WORDS:** This feature is a counter of the number of words since the last change of LAST_INCR_RESP (the number of words since the service did not change its mind about the response to deliver).

- **NORMALISED_SCORE:** At each micro-turn, the ASR score is updated: it is multiplied by the ASR corresponding to the new incoming word (see Fig. 4.3). Except from the cases where a boost takes place, the score keeps decreasing as the user speaks. To avoid penalising long sentences, the score is normalised by taking the geometric mean over the words (of course, this is an approximation as the number of inputs that formed the current ASR hypothesis is not exactly the number of words because of deletions and additions in the Scrambler). If s is the current score for n number of words, NORMALISED_SCORE $= s^{\frac{1}{n}}$.

- **TIME:** Corresponds to the duration of the current episode.

A linear model is used to represent the Q-function. First, we noticed that 21 combinations between SYSTEM_REQ and LAST_INCR_RESP are frequently visited (the others barely happen or not at all). Therefore, 21 features are defined $\delta_1,...,\delta_{21}$ where $\delta_i = 1$ if and only if the current state corresponds to the ith combination, and 0 otherwise. The rare combinations are not included in the model for two reasons. First, they require maintaining heavier models for with no real improvements over the simpler ones and second, Fitted-Q algorithm was chosen for resolving the MDP and it involves a matrix inversion that should be well conditioned. Having features that are equal to zero most of the time is not compliant with that condition.

NB_USER_WORDS is represented by three RBF functions ϕ_1^{nw}, ϕ_2^{nw} and ϕ_3^{nw} centered in 0, 5 and 10 with a standard deviation of 2, 3 and 3. In other words

$$\phi_i^{nw} = \exp\left(\frac{NB\_USER\_WORDS - \mu_i}{2\sigma_i^2}\right)$$

$$\mu_1 = 0, \ \mu_2 = 5, \ , \mu_3 = 10$$

$$\sigma_1 = 2, \ \sigma_2 = 3, \ \sigma_3 = 3$$

Similarly, NORMALISED_SCORE is also represented using two RBF functions ϕ_1^{ns} and ϕ_2^{ns} centered in 0.25 and 0.75 and with a standard deviation of 0.3 for both.

Finally, TIME is normalised so that it is near zero at the beginning of the episode and around 1 when the duration reaches 6 minutes (the maximum duration due to patience limit):

$$T = sigmoid\left(\frac{(TIME - 180)}{60}\right)$$

There is no need to use RBFs for this last feature as the Q-function is supposed to be monotonous with respect to it. The more the dialogue last, the more likely the user would hang up.

Therefore, the dialogue state is represented by the following vector

$$\Phi(s) = [1, \delta_1, \delta_2, \delta_3, \phi_1^{nw}, \phi_2^{nw}, \phi_3^{nw}, \phi_1^{ns}, \phi_2^{ns}, T]$$

6.1.2 Actions, rewards and episodes

The system can perform the action WAIT and the action SPEAK (see Chap. 5 for a description of these actions). The action REPEAT introduces more complexity to the system and therefore, it is left for future work.

At each micro-turn, the system receives a reward $-\Delta t$ which correspond to the opposite of the time elapsed since the micro-turn before. Moreover, there are two rewarding situations, where the system gets a reward of 150:

- The Scheduler retrieves a confirmation that the task corresponding to the user's requests has been accomplished to the client (happens when the user says *yes* to a confirmation question).

- The Scheduler retrieves a conflict declaration to the client. Even though the task has not been accomplished from the dialogue manager point of view, it has been from the Scheduler's side.

An episode is a portion of a dialogue that starts with an open question (where the user is supposed to utter a complete request with all the necessary slot values) and ends with either a new open question or a user hang up.

6.1.3 Fitted-Q Value Iteration

Fitted-Q iteration has already been successfully applied to dialogue management in the traditional sense (Chandramohan et al., 2010). Here it is applied to the problem of turn-taking.

where $\theta(a)$ is a parameter vector associated with action a. For the notations to be more compact, we will omit the action and refer to this parameters vector as θ. The aim of Fitted-Q algorithm is to estimate those parameters for the optimal Q-function Q^*. Recall that the Bellman optimality equation states that

$$Q^*(s,a) = \mathbb{E}[R(s,a,s') + \gamma \max_{a'} Q^*(s,a')|s,a]$$

$$Q^* = T^*Q^*$$

The operator T^* is a contraction (Bellman, 1957). As a consequence, there is a way to estimate it in an iterative way: *Value Iteration* (Banach theorem). Each new iteration is linked to the previous one as follows:

$$\hat{Q}_i^* = T^*\hat{Q}_{i-1}^*$$

However, an exact representation of the Q-function is assumed which is not possible in our case as the state space is infinite (even if we discretise the ASR score, it will still be too big). As a consequence, we adopt a linear representation of the Q-function:

$$\hat{Q}(s,a) = \theta(a)^T\Phi(s)$$

$\hat{Q}$ is the projection of Q on the space of the functions that can be written as a linear combination of the state vector's components. Let Π be the corresponding projection operator, then it can be shown that ΠT^* is still a contraction and admits a unique fixed point that can be iteratively computed as follows: $\hat{Q}_{\theta_i} = \Pi T^*\hat{Q}_{\theta_{i-1}}$. As the transition probabilities of the MDP and the reward function are not known, a sampled operator $\hat{T}$ is used instead of T. For a transition (s,a,r,s'), it is defined as

$$\hat{T}\hat{Q}(s,a) = r + \gamma \max_{a}{}' Q(s',a')$$

The Fitted-Q algorithm therefore estimates the θ vector using the iteration rule: $\hat{Q}_{\theta_i} = \hat{T}\hat{Q}_{\theta_{i-1}}$. To compute the projection, the L_2 norm is minimised:

$$\theta_i = \underset{\theta \in \mathbb{R}^p}{\arg\min} \sum_{j=1}^{N} (r_j + \gamma \max_a \theta_{i-1}(a)^T \phi s_j - \theta(a_j)\phi(s_j))$$

where N is the number of transitions in the data batch. This is a classic least square optimisation and θ_i can be computed as follows (the matrix inversion has to be performed only once and not at every iteration):

$$\theta_i = (\sum_{i=1}^{N} \phi(s_i)^T \phi(s_i))^{-1} \sum_{j=1}^{N} \phi(s_j)(r_j + \gamma \max_a \theta_{i-1}(a)^T \phi(s_j))$$

6.2 Experiment

6.2.1 Setup

The same dialogue scenarios described in Chap. 5 are used here. For learning, the noise level is fixed at 0.15. 50 learning curves have been produced with 3000 episodes each and the average curve is depicted in Fig. 6.1. The θ vectors in the Q-function model are initially zeros and they are updated every 500 episodes. There are three phases to distinguish:

1. **Pure exploration (Episodes 0-500):** The actions are taken randomly with a probability of 0.8 for choosing WAIT and 0.2 for SPEAK. Picking equiprobable actions results in the user being interrupted so often that the interesting scenarios are very rarely explored.

2. **Exploitation/exploration (Episodes 500-2500):** An ϵ-greedy policy is used with respect to the current Q-function, with $\epsilon = 0.1$.

3. **Pure exploitation (Episodes 2500-3000):** A 100% greedy policy is used.

6.2.2 Results

Three different strategies are compared:

- **Non-incremental baseline:** It corresponds to the MixIni strategy defined in Chap. 5. The user is asked to provide all the information necessary to execute her request and when there are still missing slots, the corresponding values are asked for one after another.
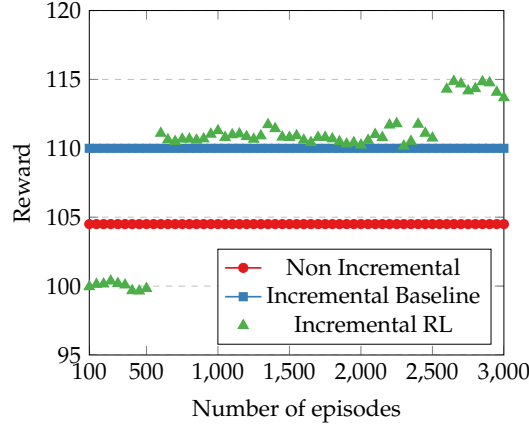
Figure 6.1: *Learning curve (0-500: pure exploration, 500-2500: exploration/exploitation, 2500-3000: pure exploitation)*

- **Incremental baseline:** MixIni+Incr from Chap. 5 is selected as an incremental baseline. It is identical to the non-incremental baseline with the difference that it is enhanced with handcrafted turn-taking rules defined in Chap. 5.

- **Incremental RL:** It corresponds to the strategy learned with reinforcement learning.

Like in Chap. 5, these strategies are compared under different levels of noise. The non-incremental and the incremental baselines have already been compared in Chap. 5. In Fig. 6.2, they are also compared to the new automatically learned strategy. The differences become clearer as the WER increases. For WER=0.3, the non-incremental baseline reaches 3 minutes, the incremental baseline goes 10 seconds faster and the incremental RL still improves it by an additional 20 seconds (17% gain in total). In terms of task completion, the non-incremental baseline drops under 70%, the incremental baseline shows a performance of 73% whereas the incremental RL keeps the ratio at a level of 80%.
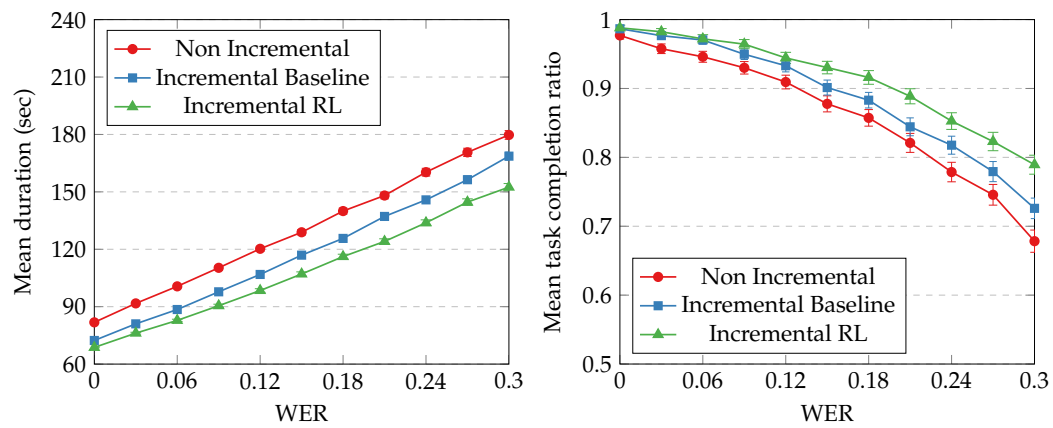
Figure 6.2: *Mean dialogue duration and task for the non-incremental, the baseline incremental and the RL incremental strategies under different noise conditions.*

Chapter 7

Experiment with real users

7.1 Protocol

7.2 Results

Chapter 8

A new reinforcement learning approach adapted to incremental dialogue

8.1 **Problem**

8.2 **Proposed approach**

8.3 **Toy problem application**

8.4 **Simulation results**

Conclusion

Conclusion here

List of Figures

List of Tables

Bibliography

(Aist et al., 2007)   G. Aist, J. Allen, E. Campana, C. G. Gallo, S. Stoness, M. Swift, and M. K. Tanenhaus, 2007.   Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods. Dans les actes de *Decalog*.

(Asri et al., 2012)   L. E. Asri, R. Laroche, and O. Pietquin, 2012. Reward function learning for dialogue management. Dans les actes de *STAIRS*.

(Auer et al., 2002)   P. Auer, N. Cesa-Bianchi, and P. Fischer, 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*.

(Auer and Ortner, 2005)   P. Auer and R. Ortner, 2005. Online-regret bounds for a new reinforcement-learning algorithm. Dans les actes de *1st Austrian Cognitive Vision Workshop*.

(Austin, 1962)   J. Austin, 1962. *How to Do Things with Words*. Oxford.

(Baumann, 2008)   T. Baumann, 2008. Simulating spoken dialogue with a focus on realistic turn-taking. Dans les actes de *Proceedings of the 5th International Workshop on Constraints and Language Processing (CSLP)*.

(Baumann, 2014)   T. Baumann, 2014. Partial representations improve the prosody of incremental speech synthesis. Dans les actes de *Proceedings of Interspeech*.

(Baumann and Schlangen, 2011)   T. Baumann and D. Schlangen, 2011. Evaluation and optimisation of incremental processors. *Dialogue and Discourse 2*, 113–141.

(Baumann and Schlangen, 2013)   T. Baumann and D. Schlangen, 2013. Open-ended, extensible system utterances are preferred, even if they require filled pauses. Dans les actes de *Proceedings of the SIGDIAL 2013 Conference*.

(Beattie, 1982)   G. Beattie, 1982. Turn-taking and interruption in political interviews: Margaret thatcher and jim callaghan compared and contrasted. *Semiotica 39*, 93–114.

(Bellman, 1957)   R. Bellman, 1957. *Dynamic Programming*. Princeton University Press.

(Bengio et al., 2003)   Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, 2003. A neural probabilistic language model. *The Journal of Machine Learning Research 3*, 1137–1155.

(Berry and Fristedt, 1985)   D. A. Berry and B. Fristedt, 1985. *Bandit problems : sequential allocation of experiments*. Chapman and Hall London.

(Bratman, 1987)   M. Bratman, 1987. *Intention, plans, and practical reason*. Harvard University Press.

(Bratman et al., 1988)   M. E. Bratman, D. J. Israel, and M. E. Pollack, 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence 4*, 349–355.

(Breslin et al., 2013)   C. Breslin, M. Gasic, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young, 2013. Continuous asr for flexible incremental dialogue. Dans les actes de *ICASSP*, 8362–8366.

(Bubeck and Cesa-Bianchi, 2012)   S. Bubeck and N. Cesa-Bianchi, 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and TrendsÂ® in Machine Learning 5*, 1–122.

(Chandramohan et al., 2011)   S. Chandramohan, M. Geist, F. LefÃ¨vre, and O. Pietquin, 2011. User simulation in dialogue systems using inverse reinforcement learning. Dans les actes de *INTERSPEECH*.

(Chandramohan et al., 2010)   S. Chandramohan, M. Geist, and O. Pietquin, 2010. Optimizing spoken dialogue management with fitted value iteration. Dans les actes de *INTERSPEECH 11th Annual Conference of the International Speech*.

(Chao and Thomaz, 2012)   C. Chao and A. L. Thomaz, 2012. Timing in multimodal turn-taking interactions: Control and analysis using timed petri nets. *Journal of Human-Robot Interaction*.

(Clark, 1996)   H. H. Clark, 1996. *Using Language*. Cambridge University Press.

(Cohen and Levesque, 1990)   P. R. Cohen and H. J. Levesque, 1990. Persistence, intention, and commitment. Dans les actes de *Intentions in Communication*.

(Collobert et al., 2011)   R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, 2011. Natural language processing (almost) from scratch. *CoRR abs/1103.0398*.

(Daubigney et al., 2013)   L. Daubigney, M. Geist, and O. Pietquin, 2013. Model-free pomdp optimisation of tutoring systems with echo-state networks. Dans les actes de *Proceedings of the SIGDIAL 2013 Conference*.

(Deng et al., 2013)   L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, 2013. Recent advances in deep learning for speech research at microsoft. Dans les actes de *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

(Dethlefs et al., 2012)   N. Dethlefs, H. W. Hastie, V. Rieser, and O. Lemon, 2012. Optimising incremental dialogue decisions using information density for interactive systems. Dans les actes de *EMNLP-CoNLL*.

(DeVault et al., 2011) D. DeVault, K. Sagae, and D. Traum, 2011. Incremental inter-pretation and prediction of utterance meaning for interactive dialogue. *Dialogue and Discourse 2*, 143–170.

(Dudley, 1939) H. Dudley, 1939. The vocoder. *Bell Labs Record 17*, 122–126.

(Duncan, 1972) S. Duncan, 1972. Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology 23*, 283–292.

(Eckert et al., 1997) W. Eckert, E. Levin, and R. Pieraccini, 1997. User modeling for spoken dialogue system evaluation. Dans les actes de *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*.

(Edlund et al., 2008) J. Edlund, J. Gustafson, M. Heldner, and A. Hjalmarsson, 2008. Towards human-like spoken dialogue systems. *Speech Communication 50*, 630–645.

(Ekeinhor-Komi et al., 2014) T. Ekeinhor-Komi, H. Falih, C. Chardenon, R. Laroche, and F. Lefevre, 2014. Un assistant vocal personnalisable. Dans les actes de *Proceedings of the 21st ConfÃ©rence sur le Traitement Automatique des Langues Naturelles (TALN)*.

(El Asri et al., 2014a) L. El Asri, H. Khouzaimi, R. Laroche, and O. Pietquin, 2014a. Ordinal regression for interaction quality prediction. Dans les actes de *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*.

(El Asri et al., 2014b) L. El Asri, R. Lemonnier, R. Laroche, O. Pietquin, and H. Khouza-imi, 2014b. NASTIA: Negotiating Appointment Setting Interface. Dans les actes de *Proceedings of LREC*.

(Fink et al., 1998) G. A. Fink, C. Schillo, F. Kummert, and G. Sagerer, 1998. Incremental speech recognition for multimodal interfaces. Dans les actes de *IECON*.

(Gales and Young, 2007) M. Gales and S. Young, 2007. The application of hidden markov models in speech recognition. *Found. Trends Signal Process. 1*.

(Gasic et al., 2013) M. Gasic, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thom-son, P. Tsiakoulis, and S. Young, 2013. Pomdp-based dialogue manager adaptation to extended domains. Dans les actes de *Proceedings of the SIGDIAL 2013 Conference*.

(Ghigi et al., 2014) F. Ghigi, M. Eskenazi, M. I. Torres, and S. Lee, 2014. Incremen-tal dialog processing in a task-oriented dialog. Dans les actes de *Fifteenth Annual Conference of the International Speech Communication Association*.

(Gravano and Hirschberg, 2011) A. Gravano and J. Hirschberg, 2011. Turn-taking cues in task-oriented dialogue. *Comput. Speech Lang. 25*(3), 601–634.

(Hastie et al., 2013) H. Hastie, M.-A. Aufaure, P. Alexopoulos, H. Cuayáhuitl, N. Deth-lefs, M. Gasic, J. Henderson, O. Lemon, X. Liu, P. Mika, N. Ben Mustapha, V. Rieser, B. Thomson, P. Tsiakoulis, and Y. Vanrompay, 2013. Demonstration of the parlance system: a data-driven incremental, spoken dialogue system for interactive search. Dans les actes de *Proceedings of the SIGDIAL 2013 Conference*.

(Ilkin and Sturt, 2011)   Z. Ilkin and P. Sturt, 2011. Active prediction of syntactic information during sentence processing. *Dialogue and Discourse 2*, 35–58.

(Jaffe and Feldstein, 1970)   J. Jaffe and S. Feldstein, 1970. *Rhythms of dialogue*. Academic Press New York.

(Johnston et al., 2014)   M. Johnston, J. Chen, P. Ehlen, H. Jung, J. Lieske, A. Reddy, E. Selfridge, S. Stoyanchev, B. Vasilieff, and J. Wilpon, 2014. Mva: The multimodal virtual assistant. Dans les actes de *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

(Jonsdottir et al., 2008)   G. R. Jonsdottir, K. R. Thorisson, and E. Nivel, 2008. Learning smooth, human-like turntaking in realtime dialogue. Dans les actes de *In Proceedings of Intelligent Virtual Agents (IVA 08*, 162–175. Springer.

(Khouzaimi et al., 2014)   H. Khouzaimi, R. Laroche, and F. Lefèvre, 2014. An easy method to make dialogue systems incremental. Dans les actes de *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

(Khouzaimi et al., 2015)   H. Khouzaimi, R. Laroche, and F. Lefèvre, 2015. Turn-taking phenomena in incremental dialogue systems. Dans les actes de *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

(Kilger and Finkler, 1995)   A. Kilger and W. Finkler, 1995. Incremental generation for real-time applications. Rapport technique, German Research Center for Artificial Intelligence.

(Kim and Banchs, 2014)   S. Kim and R. E. Banchs, 2014. Sequential labeling for tracking dynamic dialog states. Dans les actes de *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

(Konolige and Pollack, 1993)   K. Konolige and M. E. Pollack, 1993. A representationalist theory of intention. Dans les actes de *Proceedings of the 13th International Joint Conference on Artifical Intelligence (IJCAI)*.

(Laroche, 2010)   R. Laroche, 2010. *Raisonnement sur les incertitudes et apprentissage pour les systemes de dialogue conventionnels*. Thèse de Doctorat, Paris VI University.

(Laroche et al., 2008)   R. Laroche, B. Bouchon-Meunier, and P. Bretier, 2008. Uncertainty management in dialogue systems. Dans les actes de *European Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*.

(Laroche et al., 2011)   R. Laroche, G. Putois, P. Bretier, M. Aranguren, J. Velkovska, H. Hastie, S. Keizer, K. Yu, F. Jurcicek, O. Lemon, and S. Young, 2011. D6.4: Final evaluation of classic towninfo and appointment scheduling systems. Report D6.4, CLASSIC Project.

(Laroche et al., 2013)   R. Laroche, L. Roussarie, P. Baczyk, and J. Dziekan, 2013. Cooking coach spoken/multimodal dialogue systems. Dans les actes de *Proceedings of the IJCAI Workshop on Cooking with Computers*.

(Lemon et al., 2003) O. Lemon, L. Cavedon, and B. Kelly, 2003. Managing dialogue interaction: A multi-layered approach. Dans les actes de *Proceedings of the 4th SIGDIAL Workshop on Discourse and Dialogue*.

(Lemon and Pietquin, 2007) O. Lemon and O. Pietquin, 2007. Machine learning for spoken dialogue systems. Dans les actes de *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*.

(Levelt, 1989) W. J. M. Levelt, 1989. *Speaking: From Intention to Articulation*. Cambridge, MA: MIT Press.

(Levin and Pieraccini, 1997) E. Levin and R. Pieraccini, 1997. A stochastic model of computer-human interaction for learning dialogue strategies. Dans les actes de *In EUROSPEECH 97*.

(Lock, 1965) K. Lock, 1965. Structuring programs for multiprogram time-sharing online applications. Dans les actes de *AFIPS '65 (Fall, part I) Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I*.

(Louis, 2002) V. Louis, 2002. *Conception et mise en oeuvre de modèles formels du calcul de plans d'action complexes par un agent rationnel dialoguant*. Thèse de Doctorat, Université de Caen/Basse-Normandie, France.

(Lu et al., 2011) D. Lu, T. Nishimoto, and N. Minematsu, 2011. Decision of response timing for incremental speech recognition with reinforcement learning. Dans les actes de *ASRU*.

(Macherey, 2009) K. Macherey, 2009. *Statistical methods in natural language understanding and spoken dialogue systems*. Thèse de Doctorat.

(Matthias, 2009) G. M. Matthias, 2009. Incremental speech understanding in a multimodal web-based spoken dialogue system. Mémoire de Master, Massachusetts Institute of Technology.

(Mccarthy, 1979) J. Mccarthy, 1979. Ascribing mental qualities to machines. Dans les actes de *Philosophical Perspectives in Artificial Intelligence*, 161–195.

(Mccarthy and Hayes, 1969) J. Mccarthy and P. J. Hayes, 1969. Some philosophical problems from the standpoint of artificial intelligence. Dans les actes de *Machine Intelligence*, 463–502.

(McGraw and Gruenstein, 2012) I. McGraw and A. Gruenstein, 2012. Estimating word-stability during incremental speech recognition. Dans les actes de *Proceedings of the INTERSPEECH 2012 Conference*.

(Meena et al., 2013) R. Meena, G. Skantze, and J. Gustafson, 2013. A data-driven model for timing feedback in a map task dialogue system. Dans les actes de *Proceedings of the SIGDIAL 2013 Conference*.

(Meyer, 2006) G. Meyer, 2006. *Formalisation logique de préférences qualitatives pour la sélection de la réaction d'un agent rationnel dialoguant*. Thèse de Doctorat, Université de Paris XI.

(Mohamed et al., 2009) A. Mohamed, G. Dahl, and G. Hinton, 2009. Deep belief networks for phone recognition. *NIPS 22nd workshop on deep learning for speech recognition*.

(Mori, 1970) M. Mori, 1970. The uncanny vally. Dans les actes de *Energy*.

(Newell, 1980) A. Newell, 1980. The knowledge level. *AI Magazine 2*, 1–20.

(Paek and Pieraccini, 2008) T. Paek and R. Pieraccini, 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communication 50*, 716–729.

(Pakucs, 2003) B. Pakucs, 2003. Towards dynamic multi-domain dialogue processing. Dans les actes de *8th European Conference on Speech Communication and Technology, EUROSPEECH - INTERSPEECH*.

(Pieraccini and Huerta, 2005) R. Pieraccini and J. Huerta, 2005. Where do we go from here? research and commercial spoken dialog systems. Dans les actes de *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*.

(Pietquin and Hastie, 2013) O. Pietquin and H. Hastie, 2013. A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review 28*.

(Plátek and Jurčíček, 2014) O. Plátek and F. Jurčíček, 2014. Free on-line speech recogniser based on kaldi asr toolkit producing word posterior lattices. Dans les actes de *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

(Raux and Eskenazi, 2007) A. Raux and M. Eskenazi, 2007. A multi-layer architecture for semi-synchronous event-driven dialogue management. Dans les actes de *ASRU*.

(Raux and Eskenazi, 2008) A. Raux and M. Eskenazi, 2008. Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. Dans les actes de *SIGDIAL*.

(Raux and Eskenazi, 2009) A. Raux and M. Eskenazi, 2009. A finite-state turn-taking model for spoken dialog systems. Dans les actes de *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, 629–637.

(Rosenthal et al., 2013) S. Rosenthal, D. Bohus, E. Kamar, and E. Horvitz, 2013. Look versus leap: Computing value of information with high-dimensional streaming evidence. Dans les actes de *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*.

(Sacks et al., 1974) H. Sacks, E. A. Schegloff, and G. Jefferson, 1974. A simplest systematics for the organization of turn-taking for conversation. *Language 50*, 696–735.

(Sadek, 1991)   M. D. Sadek, 1991.   *Attitudes Mentales et Interaction Rationnelle : vers une théorie formelle de la Communication*. Thèse de Doctorat, Université de Rennes I, France.

(Schatzmann et al., 2007)   J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. Dans les actes de *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*.

(Schatzmann et al., 2006)   J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review 21*.

(Schegloff, 1968)   E. A. Schegloff, 1968. Sequencing in conversational openings. *American Anthropologist 70*, pp. 1075–1095.

(Schlangen and Skantze, 2011)   D. Schlangen and G. Skantze, 2011. A general, abstract model of incremental dialogue processing. *Dialogue and Discourse 2*, 83–111.

(Searle, 1969)   J. Searle, 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, UK.

(Searle, 1968)   J. R. Searle, 1968. Austin on locutionary and illocutionary acts. *The Philosophical Review*.

(Selfridge and Heeman, 2010)   E. Selfridge and P. A. Heeman, 2010. Importance-driven turn-bidding for spoken dialogue systems. Dans les actes de *ACL*, 177–185.

(Selfridge et al., 2011)   E. O. Selfridge, I. Arizmendi, P. A. Heeman, and J. D. Williams, 2011. Stability and accuracy in incremental speech recognition. Dans les actes de *Proceedings of the SIGDIAL 2011 Conference*.

(Selfridge et al., 2012)   E. O. Selfridge, I. Arizmendi, P. A. Heeman, and J. D. Williams, 2012. Integrating incremental speech recognition and pomdp-based dialogue systems. Dans les actes de *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

(Selfridge and Heeman, 2012)   E. O. Selfridge and P. A. Heeman, 2012. A temporal simulator for developing turn-taking methods for spoken dialogue systems. Dans les actes de *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

(Sidner et al., 2013)   C. Sidner, T. Bickmore, C. Rich, B. Barry, L. Ring, M. Behrooz, and M. Shayganfar, 2013. Demonstration of an always-on companion for isolated older adults. Dans les actes de *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

(Skantze and Johansson, 2015)   G. Skantze and M. Johansson, 2015. Modelling situated human-robot interaction using iristk. Dans les actes de *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

(Skantze and Schlangen, 2009) G. Skantze and D. Schlangen, 2009. Incremental dialogue processing in a micro-domain. Dans les actes de *ACL*.

(Strombergsson et al., 2013) S. Strombergsson, A. Hjalmarsson, J. Edlund, and D. House, 2013. Timing responses to questions in dialogue. Dans les actes de *INTERSPEECH Proceedings*.

(Sullivan, 1947) H. Sullivan, 1947. *Conceptions of Modern Psychiatry*. Norton.

(Sutton and Barto, 1998) R. S. Sutton and A. G. Barto, 1998. *Reinforcement Learning, An Introduction*. The MIT Press, Cambridge, Massachusetts, London, England.

(Tabet and Boughazi, 2011) Y. Tabet and M. Boughazi, 2011. Speech synthesis techniques. a survey. Dans les actes de *Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on*.

(Tanenhaus et al., 1995) M. K. Tanenhaus, M. J. Spivey-Knowlton, K. M. Eberhard, and J. C. Sedivy, 1995. Integration of visual and linguistic information in spoken language comprehension. *Science 268*, 1632–1634.

(Walker et al., 1997) M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella, 1997. Paradise: a framework for evaluating spoken dialogue agents. Dans les actes de *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*.

(Wang et al., 2014) Z. Wang, H. Chen, G. Wang, H. Tian, H. Wu, and H. Wang, 2014. Policy learning for domain selection in an extensible multi-domain spoken dialogue system. Dans les actes de *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

(Watkins, 1989) C. J. C. H. Watkins, 1989. *Learning from Delayed Rewards*. Thèse de Doctorat, King's College.

(Williams and Young, 2007) J. D. Williams and S. Young, 2007. Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*.

(Wirén, 1992) M. Wirén, 1992. *Studies in Incremental Natural Language Analysis*. Thèse de Doctorat, Linkoping University, Linkoping, Sweden.

(Wlodarczak and Wagner, 2013) M. Wlodarczak and P. Wagner, 2013. Effects of talk-spurt silence boundary thresholds on distribution of gaps and overlaps. Dans les actes de *INTERSPEECH Proceedings*.

(Yngve, 1970) V. H. Yngve, 1970. On getting a word in edgewise. Dans les actes de *CLS-70*.

(Yuan et al., 2006) J. Yuan, M. Liberman, and C. Cieri, 2006. Towards an integrated understanding of speaking rate in conversation. Dans les actes de *INTERSPEECH Proceedings*.

(Zhao et al., 2015) T. Zhao, A. W. Black, and M. Eskenazi, 2015. An incremental turn-taking model with active system barge-in for spoken dialog systems. Dans les actes de *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.