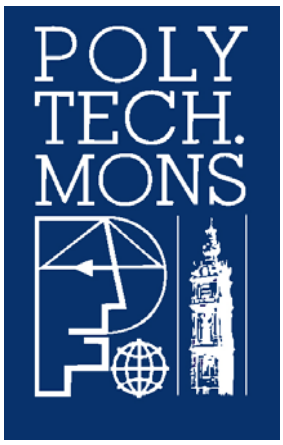
A Framework for Unsupervised Learning of Dialogue Strategies

Ir. Olivier Pietquin

Supervisor: Prof. Dr. Ir. Thierry Dutoit



Dissertation submitted to the Faculté Polytechnique de Mons
for the degree of Doctor of Philosophy in applied sciences

Faculté Polytechnique de Mons

Dissertation soumise à la Faculté Polytechnique de Mons
en vue de l'obtention du grade de docteur en sciences appliquées par

Ir. Olivier Pietquin

A Framework for Unsupervised Learning of Dialogue Strategies

Membres du jury:

Professeur Damien Macq, Faculté Polytechnique de Mons, Président
Dr. Roberto Pieraccini, IBM Research,Yorktown Heights, New York, USA
Professeur Steve Renals, University of Edinburgh, United Kingdom
Dr. Christophe d'Allessandro, CNRS-LIMSI, France
Professeur Pierre Manneback, Faculté Polytechnique de Mons
Professeur Marc Pirlot, Faculté Polytechnique de Mons
Professeur Thierry Dutoit, Faculté Polytechnique de Mons, Promoteur
Professeur Jean Hanton, Faculté Polytechnique de Mons, Doyen

Thèse préparée au laboratoire de Théorie des Circuits et Traitement du Signal de la Faculté Polytechnique de Mons et dans le groupe 'Speech and Hearing' de l'Université de Sheffield.

"He fell asleep murmuring 'Sanity is not statistical,'
with the feeling that this remark contained in it a profound wisdom."

George Orwell – "1984"

To those who still can disappoint me …

Abstract

Nowadays Human-Computer Interfaces (HCI) are widely studied and become one of the major interests among the scientific community. Indeed, more and more electronic devices surround people in their day-to-day life. Yet this exponential incursion of electronics in homes and offices is not only due to its ability to ease the achievement of common and boring tasks or the continuously decreasing prices but also because more and more user-friendly interfaces make it easier to use.

Last decades, the fields of Automatic Speech Recognition (ASR), Text-To-Speech (TTS) synthesis and Natural Language Processing (NLP) knew lots of progresses. It is now allowed to think about building real Spoken Dialogue Systems (SDS) interacting with human users through voice interactions. Speech often appears as a natural way to interact for a human being and it provides potential benefits such as hand-free access to machines, ergonomics and greater efficiency of interaction. Yet, speech-based interfaces design has been an expert job for a long time. It necessitates good skills in speech technologies and low-level programming. Moreover, rapid design and reusability of previously designed systems are almost impossible. For these reasons, but not only, people are less used to interact with speech-based interfaces which are therefore thought as less intuitive since they are less widespread.

This dissertation proposes to apply Artificial Intelligence (AI) techniques to the problem of SDS prototype design. Although Machine Learning and Decision Making techniques have already been applied to SDS optimisation, no real attempt to use those techniques in order to design a new system from scratch has been made. In this dissertation, we propose some novel ideas in order to achieve the goal of easing the design of Spoken Dialogue Systems and allow novices to have access to voice technologies.

To do so, a framework for simulating and evaluating dialogues and learning optimal dialogue strategies is proposed. The simulation process is based on a probabilistic description of a dialogue and on the stochastic modelling of both artificial NLP modules composing a SDS and the user. This probabilistic model is based on a set of parameters that can be tuned thanks to prior knowledge (to allow design from scratch) or learned from data (to allow system enhancement). The evaluation is part of the simulation process and is based on objective measures provided by each module. Finally, the simulation environment is connected to a learning agent using the supplied evaluation metrics as an objective function in order to generate an optimal behaviour for the SDS.

Acknowledgement

"What do you want to do when you're grown up?"

In my younger days, considering that other animals deserved more attention than those of my kind, 'veterinary surgeon' was my main response. Hope made me change my mind and I finally often answered: "When I'm big, I want to be a researcher". Although I don't really know if I am a big boy now, this wish came true on the 1st of September 1999. I would like to thank people who made this possible.

First, I'd like to address my thanks to Professor Henri Leich who hired me in his Circuit Theory and Signal Processing laboratory and kept his promises about giving me the time to finish this thesis. I also want to thank Professor Joël Hancq who succeeded Professor Leich at the head of the lab and above all Professor Thierry Dutoit who consented to supervise my researches for four years. He has been a real support and an example whether in a scientific, psychological or simply human way.

During six months, I had the chance to work with the Speech and Hearing laboratory of the University of Sheffield (UK) and to meet Professor Steve Renals. Our weekly meetings, his trust and his objective advices were preciously helpful.

Above all, my close family deserves all my gratitude. My mother and my stepfather made so many sacrifices during all those years. A particular thought goes to my grandmother who shortened her sleeping time everyday as well as to my grandfather who disappeared too early and transmitted me his enthusiasm for technology. All of them believed in me more than I sometimes did myself.

Even though I didn't meet them as much as they meet each other, my early days colleagues, Laurent, Christophe, Erhan, Jean-Marc, Stéphane, Michel and others welcomed me among them. Some of them became very good mates and I will probably never forget Erhan's great wisdom that made me laugh so many times. A special thank goes to Jean-Marc who set up this project in which I learned lot more than scientific issues and I would also like to thank Geoffrey and Richard for their help.

There are people you can't classify. They cross your life like UFOs, surprise you and are like milestones that make you want to see what is after. I want to thank the too few flying saucers I met during those years.

Finally, I would like to thank the 'Région Wallonne' and the 'Direction Générale des Technologies, de la Recherche et de l'Energie' (DGTRE) for their financial support as well as the 'Faculté Polytechnique de Mons' (FPMs) and the Babel Technologies society for their administrative support.

Content

Table of Figures

List of Tables

Table of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
ATN	Augmented Transition Network
AV	Attribute-Value
AVM	Attribute Value Matrix
BBN	Bolt, Beranek, and Newman
BIC	Bayesian Information Criterion
BN	Bayesian Network
CFG	Context-Free Grammar
CGT	Conversational Game Theory
CL	Confidence Level
CMU	Carnegie Mellon University
CPD	Conditional Probability Distribution
CPT	Conditional Probability Table
DARPA	Defence Advanced Research Projects Agency
DBN	Dynamic Bayesian Network
DE	Discourse Entity
DM	Dialogue Manager
DOD	Department Of Defense
DP	Dynamic Programming
DSP	Digital Signal Processing
DTW	Dynamic Time Warping
EM	Expectation-Maximization
FSG	Finite State Grammar
GMM	Gaussian Mixture Model
HCI	Human-Computer Interface
HMD	Human-Machine Dialogue
HMM	Hidden Markov Model
HTML	HyperText Markup Language
KR	Knowledge Representation
LDS	Linear Dynamic System
LM	Language Model
LPC	Linear Prediction Coding
MAP	Maximum a Posteriori
MDP	Markov Decision Process
MFCC	Mel Frequency Cepstral Coefficients
MIT	Massachusset Institute of Technology
ML	Maximum Likelihood
MLR	Multivariate Linear Regression
MMDS	MultiModal Dialogue System
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
PADIS	Philips Automatic Directory Information System
PARADISE	PARAdigm for DIalogue Systems Evaluation
PHP	Hypertext PreProcessor
PLP	Perceptual Linear Prediction

POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
SAMPA	Speech Assessment Methods Phonetic Alphabet
SCM	Substitution Cost Matrix
SDC	Systems Development Corporation
SDS	Spoken Dialogue System
SRI	Stanford Research Institute
STN	State-Transition Network
SUR	Speech Understanding Research
TD	Temporal Distance
TTS	Text-To-Speech
UM	User Model
VAD	Voice Activity Detector
VoiceXML	Voice eXtensible Markup Language
W3C	World Wide Web Consortium
WER	Word Error Rate
WK	World Knowledge
WOZ	Wizard of Oz
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XTND	XML Transition Network Definition

First Part

~

INTRODUCTION

Articulated language is one of the main human being's characteristics. This is probably why the faculty of expressing sequences of ideas and concepts using articulated language is often presumptuously confused with intelligence. As well, the ability of understanding and producing more or less coherent answers to spoken utterances implicitly defines different degrees of intelligence.

> *"… it is highly deserving of remark, that there are no men so dull and stupid, not even idiots, as to be incapable of joining together different words, and thereby constructing a declaration by which to make their thoughts understood …"*
>
> René Descartes (1637) – "Discourse on the method"

Our definition of intelligence also deals with the facility of learning. The faster a child learns, the more intelligent he is. The aptitude to understand and associate concepts, to generalize from experience to unseen situations and to act rationally by applying knowledge are all part of what we think to be intelligence but are closely related to the definition of learning.

Because we like to think that intelligence is peculiar to mankind, philosophers stated that machines could never be as clever as to be able to speak coherently by their own and science fiction authors' most fearsome creatures are machines which learning rate outperforms ours. But we are on the way …

> *"HAL 9000: I honestly think you ought to calm down; take a stress pill and think things over."*
>
> Arthur C. Clarke (1968) – "2001: A space odyssey"

Chapter I:
Why and How?

I.1. Motivations

Nowadays Human-Computer Interfaces (HCI) are widely studied and become one of the major interests among the scientific community. Indeed, more and more electronic devices are surrounding people in their day-to-day life. Yet this exponential incursion of electronics in homes and offices is not only due to its ability to ease the achievement of common and boring tasks or the continuously decreasing prices but also because more and more user-friendly interfaces make it easier to use. Personal computers and mobile phones are some of the best examples of devices that have an increasing success because they are offering powerful capabilities to any user with a minimum of training. More and more devices like fridges, washing machines or microwave ovens are being equipped with LCD screens allowing access to additional functionalities with less effort, VCR and audio/video receivers provide 'on-screen' menu display etc. All those new interfaces enable non-expert or non-trained users to access new technologies or to make easier use of options provided by a given device.

Due to last decades' progresses in the field of speech technologies, like Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) synthesis and in the field of Natural Language Processing (NLP), voice-enabled interfaces and Spoken Dialogue Systems (SDS) are hopefully going to become growingly common. Since speech is the more natural way to interact for people, speech-based interfaces seem to be a better option to interact with common devices than learning to use visual or even mechanical interfaces. Nevertheless, until now, only few working systems were released to real users and didn't really prove to outperform standard interfaces in terms of task completion and ergonomics. This causes users to *a priori* dislike speech-based interfaces after having used one once. For instance, voice dialling is a common selling point of recent mobile phones and is almost never used by their owner.

This observation is of course partly due the statistical nature of speech technologies, making them error prone, but not only. Indeed, it is also due to the management of error corrections, to the lack of immediate feedback of pure speech-based interface and above all to the lack of flexibility. Indeed, each user can have a different way to interact with a visual interface while a voice-enabled interface is commonly more rigid. This potential inflexibility is generally the cost to pay for a lower error rate and a better task completion

rate because constraining possible speech entries enhances the speech processing results. Moreover, speech-based interaction is a linear sequential process, which is by nature less flexible than communication through a visual interface that offers lots of possibilities at a same time. Finally, since speech is a more natural way to interact, beginner users' demands are generally more complex than what the SDS is designed for and even more complex that what they could obtain from a graphical interface. Users do not adapt themselves to the interface (like they do for graphical interfaces) and even if they do, they conclude that it is less powerful than a graphical interface while it is not always the case.

Besides, over the last decade, the web-based technologies enabled non-expert people to develop themselves simple visual interfaces thanks to interpreted scripting languages such as HTML (HyperText Markup Language) and more generally XML (eXtensible Markup Language). Very powerful graphical design tools made even easier to build web sites and new comers to computer science could contribute to the growth of the Internet. Thus, visual interfaces for information retrieval, form filling or database access became widely used. Those tools also contributed to the standardization of interfaces. On another hand, speech-based interfaces design has been an expert job for a long time. It necessitates good skills in speech technologies and low-level programming. Moreover, rapid design and reusability of previously designed systems are almost impossible. For these reasons also, people are less used to interact with speech-based interfaces which are therefore thought as less intuitive since they are less widespread.

The following text addresses both the usability and design issues, which are finally close to each other. Indeed, lack of flexibility is of course due to the nature of the speech-based communication but also because the design of SDS is more complex, more task-dependent and user-dependent and relies on lots of different factors (like the ASR performance, noise etc.). In the purpose of an easier design and a better usability of voice-enabled interfaces, this thesis describes methods for evaluation, simulation and optimal behaviour learning of SDS.

I.2. Practical Context

This thesis has been realized in the framework of a 'First Europe' project funded by the 'Direction Générale des Technologies, de la Recherche et de l'Energie' (DGTRE) of the Région Wallonne in Belgium. Practically, the 'First Europe' framework implies the collaboration of two European universities (the 'Faculté Polytechnique de Mons' (FPMs) in Belgium and the University of Sheffield in United Kingdom in this case) in which the researcher has to spend at least six months and a European company (Babel Technologies from Belgium in this case). It aims at creating connections among European academic institutions but also between academic institutions and the industry. This desired connection between academics and industries induces consequences in the approach of the funded researches and it is a general will of the 'Région Wallone' to fund applied research projects and not fundamental researches.

This research was realised both in the 'Théorie des Circuits et Traitement du Signal' (TCTS) laboratory of the FPMs and the SPeech and Hearing (SPandH) group of the University of Sheffield. These labs have a strong background in fundamental signal processing and mainly speech processing. Particularly, the TCTS department, where most of the researches were realised, was mainly split into two teams focusing their researches on Digital Signal Processing (DSP) for speech recognition and speech synthesis and was not directly involved in dialogue management. On another hand, the associate company

Babel Technologies was specialised in SDK development and selling more than end-user products.

According to those practical constraints, the specifications of the project were drawn and can be summarised as follow: development of an application dedicated to build task-oriented spoken dialogue systems mainly for information retrieval, form filling and database access. The application should help a designer (expert or not) to create optimised prototypes of dialogue systems from scratch and therefore should not take advantage of data collections obtained by releasing prototypes to users. This thesis reports researches leaded in the aim of achieving this task. The influence of each scientific team in which this research has been realised leaded not to consider dialogue design as only a very high level process but to take into account properties of all the modules concerned.

I.3. Contributions

The behaviour of a dialogue system is determined by its strategy. The strategy is contained in a dialogue management (DM) module, which can be considered as the brain of the system while the remaining modules are essentially NLP modules. Designing a dialogue strategy, and so, defining the scheduling of voice interactions, is probably the most delicate task of the SDS design process (given the DSP and NLP modules, of course). Some researches in dialogue metrics and automatic strategy learning attempted to solve the problem of dialogue strategy design but results remain difficult to use in practice and it is still an expert job. They are usually used for evaluation and sometimes enhancement of already existing systems more than for designing new strategies from scratch.

As said before, this thesis addresses the problem of designing task-oriented spoken dialogue system prototypes merely according to knowledge about the task. This supposes that the designer doesn't dispose of data collected thanks to previous prototypes or so-called 'Wizard of Oz' simulations. These restrictions are made in order to possibly give access to SDS design to non-expert designers.

In the proposal made in this dissertation, we describe a parametric model for generating optimal spoken dialogue strategies. That is the better sequence of interactions that will lead to the achievement of the task in terms of performance and user satisfaction. In this framework, the model relies on a limited set of parameters that can be estimated thanks to prior knowledge (that means, without data collection) but all the mechanisms for learning those parameters from collected data or even online learning are also available, making the application not restricted to the simple case of *a priori* knowledge. The application can then be used in the aim of improvement of already existing systems.

Like in previous researches, it is proposed to simulate dialogues in order to evaluate them and also to provide artificial data points allowing optimal strategy learning. Yet, the simulation process described in this text is different from previous work in the domain and is based on a more realistic description of the SDS environment. A probabilistic description of a dialogue serves as a base for the simulation process. The proposed simulation environment models the DSP and NLP modules surrounding the dialogue manager (DM) and the system's user in a stochastic way. Where previous works have a quite simple joined error model for Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) modules, this text proposes to adapt error models to dialogue state according to task knowledge. Indeed, SDS are often parameterised

differently at each dialogue step in order enhance the performance (different language models and vocabulary sizes for the ASR system, for instance). On another hand, it is suggested to model system's users thanks to Bayesian Networks (BN). This representation allows the inclusion of task knowledge, dialogue history and other parameters in the user's decision-making process. Moreover, the BN framework is suitable for combining prior knowledge and parameter learning. BNs also allow goal inferring and user adaptation, which means that it can also be used as part of the dialogue systems. Each module of the simulation environment also provides objective metrics allowing the evaluation of every interaction.

The simulation environment is then connected to a learning agent that can take a combination of the evaluation metrics as an objective function to produce an optimal strategy according to the properties of each module and to the task. In this dissertation, the Reinforcement Learning (RL) paradigm is described, since it is an unsupervised learning technique able to learn from interactions with an environment (real or simulated). Although RL has already been applied to dialogue management, in this text new reinforcement signals and state spaces are proposed in the framework of factored Markov Decision Processes (MDPs). Nevertheless, other decision-making techniques can be used since the simulation process is sufficiently generic.

I.4. Plan

The First Part of this text, including this chapter, is mainly introductory and will define techniques used in the field of dialogue systems and AI used later in the text. The area of Human-Machine Dialogues (HMD) is quite large and relies on lots of other research fields going from low level signal processing like used in the ASR process to higher level based on the manipulation on concepts like in Natural Language Understanding (NLU) or Knowledge Representation (KR). The second chapter will describe those fields. In chapter III, the general Reinforcement Learning (RL) and Bayesian Network (BN) frameworks are described.

In the Second Part, the dialogue simulation process is explained. Chapter IV reports a general probabilistic framework for describing HMDs. This approach permits to define what are the probabilities to evaluate for dialogue simulation purpose. Chapter V exposes two different goal directed user models developed in the framework of this thesis. They are both probabilistic but the second is based on the BN framework and produces more realistic and powerful results. Eventually, chapter VI describes an attempt to simulate the speech and natural language processing modules behaviour and their possible errors.

Strategy learning techniques will be discussed in the Third Part of this dissertation. Chapter VII describes the definition of the dialogue problem in the framework of factored MDPs in the purpose of automatic learning of dialogue strategies. Illustrating examples are depicted in Chapter VIII. These examples demonstrate the ability of the learning algorithm to adapt to different situations simulated by the probabilistic simulation environment defined in previous chapters.

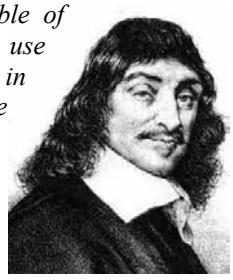
Conclusion and further works, including possible uses of the techniques described in this text, are exposed in the Fourth Part.

Chapter II:
Human - Machine Spoken Dialogue: State of the Art

II.1. A Bit of History

Machines able to produce speech are not a recent idea. With the ancient oracles, the thought that inanimate objects could speak was born. Although they did not always agree on the fact of that being possible, the 'mathematicians-philosophers' of the 17th and 18th centuries, contemporaries of first computational machines, also thought about it. For example, Descartes declared that even if machines could speak, their discourse could not be coherent.

"... if there were machines bearing the image of our bodies, and capable of imitating our actions as far as it is morally possible [...] they could never use spoken words or other signs arranged in such a manner as is competent to us in order to declare our thoughts to others: for we may easily conceive a machine to be so constructed that it emits vocables, and even that it emits some correspondent to the action upon it of external objects which cause a change in its organs [...] but not that it should arrange them variously so as appositely to reply to what is said in its presence ..."



René Descartes (1637) – "Discourse on the method"

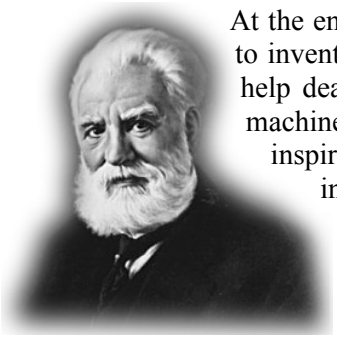
More than a hundred years later, Leonhard Euler was more optimistic than his peer and said:

"It would be a considerable invention indeed, that of a machine able to mimic speech, with its sounds and articulations. I think it is not impossible."

Leonhard Euler (1761)

Nevertheless, one of his contemporaries, Wolfgang von Kempelen (shown on the left drawing), built the first 'talking machine' and described it precisely in a book published in 1791 [von Kempelen, 1791]. His machine was not only able to produce speech sounds but also words and even short sentences in Latin, French and Italian languages. It was inspired by the model of the human vocal tract built in 1779 by Christian Kratzenstein, which was only able to produce vowel sounds. Different versions of the 'von Kempelen's Machine' were built

during the 19th and even 20th century.



At the end of the 19th century, Alexander Graham Bell (left picture) decided to invent a machine that could transcribe spoken words into text in order to help deaf people. Although Bell was unsuccessful in the invention of this machine (the 'phonoautograph'), it was while working on it that he had the inspiration for what would become the telephone in 1878. This important invention and the progresses in electrical engineering made in the first half of the 20th century opened the way of functional (and not physical) speech sound production by electrical means.

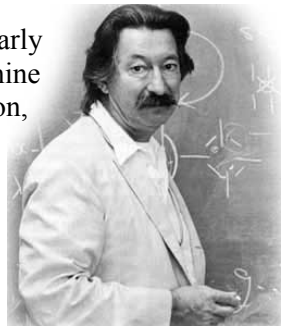
In 1939, Homer Dudley, a research physicist at Bell Labs, demonstrated his VODER at the World's Fair in New York [Science News Letter, 1939]. The VODER was still a manually controlled speech synthesizer as was the 'von Kempelen's Machine' and highly trained technicians that operated analogue and continuous controls were necessary to manipulated it [Dudley et al, 1939].

In the mean time, the Elmwood Button Company produced in 1922 a toy dog (the 'Radio Rex') that made history as the first successful entry into the field of speech recognition. It used an electromagnet sensitive to sound patterns containing acoustic energy around 500 Hz, such as the vowel in 'Rex', and gave the illusion that the celluloid dog responded to its name.

Since the second half of the 20th century, the parallel and interleaved development of several research fields contributed to make the old dream of talking machines come true. It started with von Kempelen's physical speech synthesis attempts and Bell's thought of Automatic Speech Recognition (ASR) but half a century of scientific, linguistic, psychological and even philosophical researches were needed.

In the late 1940's, the U.S. Department of Defense (DOD), in an attempt to speed up the processing of intercepted Russian messages, sponsored the first researches in speech recognition. In 1952, the Bell Laboratories developed a system that could successfully recognize spoken digits transmitted by phone with an accuracy of 98% with speaker adaptation [Davis et al, 52]. In 1959, at the Lincoln Lab of the Massachusset Institute of Technology (MIT), Forgie and Forgie developed a speaker-independent system able to recognize vowels with an accuracy of 93% [Forgie & Forgie, 1959]. Ben Gold, from the same lab, demonstrated in 1966 a system able to match a spoken utterance to a list of 50 words and to associate a score with the recognition result. The system performed isolated word recognition with an accuracy of 83%.

It is also during the 1950's and 1960's at the MIT that Noam Chomsky developed his theories about linguistics and computational grammars [Chomsky, 1956], [Chomsky, 1965] and built the foundation of Natural Language Processing (NLP).



The concept of Artificial Intelligence (AI) was born at Bell Labs in the early 1950's. It is linked with two landmark papers on chess playing by machine written by the father of information theory, Claude E. Shannon [Shannon, 1950 a]. Nevertheless, it is in 1966 that Joseph Weizenbaum (from MIT, shown on the right-hand picture) released ELIZA (so named after the Eliza Doolittle character in the play "Pygmalion"), the first artificial intelligence program simulating human conversation that successfully (but informally) passed the Turing test [Weizenbaum,

1966]. According to this test, introduced by Allan Turing in 1950 [Turing, 1950], if an interrogator is having a conversation with both a computer and another person through a terminal and cannot find out which is the machine by asking questions, then the machine is intelligent. ELIZA mimicked the behaviour of a psychologist and was tested on the non-technical staff of the MIT AI Lab. Weizenbaum was shocked by the results of his experiment as lots of the users spent hours revealing their personal problems to the program.

In 1968, in their sci-fi masterpiece '2001: A Space Odyssey', Arthur C. Clarke and Stanley Kubrick created HAL9000, a computer that could hold a conversation, think and adapt its behaviour.

During the 1970's, the Defense Advanced Research Projects Agency (DARPA), a unit of the DOD, funded a five-year project called the Speech Understanding Research (SUR) program. It aimed at developing a computer system that could understand continuous speech and major research groups were established at MIT's Lincoln Lab, Carnegie Mellon University (CMU), Stanford Research Institute (SRI), System Development Corporation (SDC) and Bolt, Beranek, and Newman (BBN). It leaded to the HARPY and DRAGON speech recognition systems but none of them met the project's objectives as their vocabularies were too small and their error rates too high. Anyway, it is during those years that James and Janet Baker (from CMU) and Frederick Jelinek (from IBM) began applying the statistical pattern-matching framework named Hidden Markov Models (HMM) to speech recognition [Jelinek, 1976]. HMMs were a real breakthrough in the area of pattern matching and were widely studied in the early 1970's by Leonard Baum from Princeton University [Baum, 1972].

With HMMs and, more generally speaking, statistical pattern matching, cognitive science entered in the very important era of data-driven or corpus-based techniques. They are still now commonly used in all fields of NLP including ASR, Natural Language Understanding (NLU), Natural Language Generation (NLG) and Text-to-Speech (TTS) synthesis.

Thanks to the increase of the processing power, the 1970's were a decade of great progresses in the field of Digital Signal Processing (DSP) and especially for speech processing like Rabiner and Shafer described in their highly influential book [Rabiner & Shafer, 1978]. It is also in the course of this decade that the structure of human discourse began to be the object of formal investigations in the purpose of making Human-Computer Interfaces (HCI) more 'user-friendly'. In 1974, Barbara Grosz from SRI started studying the structure of dialogues in collaborative tasks [Grosz, 1974], which was a step further than Chomsky's syntactic studies.

The 1980's saw the creation of the first companies incorporating research results into products (Dragon Systems in 1982, Speechworks in 1984). The collection of large standard corpora for speech recognition systems training and testing began. Lots of NLU researches were conducted during the 1980's. For example, Barbara Grosz and Candace Sidner developed the theory of 'centering' [Grosz & Sidner, 1986] that aimed to formalize the way a human follows the focus of a conversation. James Allen applied statistical pattern-matching techniques usually applied in speech recognition to semantic parsing of natural language [Allen, 1987].

In the first half of the 1990's, hybrid methods combining Artificial Neural Networks (ANN) and HMMs were successfully used in large speech recognition systems [Bourlard & Morgan, 1994]. In 1994, Nuance Communications, a spin off of the SRI, is founded and is still now one of the leaders in the domain. And it is only in the second half of this decade that development of complete Spoken Dialogue Systems (SDS) including speech recognition, speech synthesis, NLU and dialogue management started to emerge. In 1996,

Charles Schwab is the first company to propose a fully automatic telephone service. It allows for up to 360 simultaneous customers to call in and get quotes on stock and options.

II.2. Generalities about Human-Machine Dialogue

Nowadays, thanks to half a century of intensive researches, interacting with machines becomes amazingly common. People born during the two last decades are used to communicate with computers, answering machines, video games, etc. However, a real Human-Machine Dialogue (HMD) takes place in particular applications that will be described in the following.

Before entering into details about Spoken Dialogue Systems (SDS), it is important to have some fundamental notions about dialogue in general and HMD in particular. This section will give some simple definitions, describe some types of dialogue applications, define different levels of communication coming up during spoken interactions, and then the general architecture of generic dialogue systems.

II.2.1. Definitions

In the rest of this dissertation, a *dialogue* will be referred to as an interaction between two *agents* based on sequential turn taking. In most of the cases, this interaction is *goal-directed* and both agents cooperate in order to achieve an aim. In the case of a HMD one of the agents is a human *user* while the other is a computer. In the particular case in which the interaction uses speech as the main communication mean, the computer implements a *Spoken Dialogue System* (SDS) while a system using several means of communication is referred to as a *Multimodal Dialogue System* (MMDS). When the HMD is dedicated to the realisation of a particular task (or set of tasks) it is called a *task-oriented* dialogue system. When one of the agents is an SDS, the dialogue consists of a sequence of *utterances* exchanged at each turn. A *spoken utterance* is the acoustic realisation of the *intentions* or *concepts* one of the agents wants to communicate to the other and is expressed as a *word* sequence.

II.2.2. Applications

Voice-enabled interfaces are now surrounding us and are more or less sophisticated. Indeed, it is for example possible to activate voice dialling on your mobile phone, dictate documents to a text editing tool, open and control applications on the desktop of a computer, call an information providing system to ear a summary of your e-mails or to know about the weather in a particular area, your call can also be automatically routed and monitored… In general, HMD systems can be classified according to the application type they are dedicated to and to the complexity of the task.

Form Filling

The form filling applications are also referred to as frame-based, frame-driven or slot-filling applications in the literature. It is probably the simplest application category and was one of the first to be investigated [Bobrow et al, 1977]. This type of applications implies that the user has to provide values for a set of specific fields of a form. The purpose of the HMD system is to collect all the values.

Information Retrieval

In this case, the purpose of the system is to retrieve some specific information in a knowledge base and to provide it to the human user. Such a system should, for instance, retrieve a particular record in a database according to its user's will. Flight or train ticket booking engines, automatic book or computer retailing systems are some examples [den Os et al, 1999]. Such an application could also allow using mobile devices for accessing an information management system in a warehouse [Bagein *et al*, 2003]. The system could also retrieve pertinent documents about specific topics among websites [Lau et al, 1997] or provide weather forecast for a specific region [Zue et al, 2000]. Automatic call steering can also be classified in this category, as the system's purpose is to route calls to operators able to provide the desired information [Durston et al, 2001]. All those tasks imply to collect a certain amount of relevant pieces of information about what has to be retrieved by querying the user [Bennacef et al, 1994].

Problem Solving

Troubleshooting is typically the kind of task for which the user needs his problem to be solved (getting his/her device working properly) and for which collaborative dialogue will take place [Smith *et al*, 1992]. In this case, the system has to guide the user in order to collect information about what is going wrong but its role is also to ensure that the user will perform proper actions in order to fix the device. Consequently, the particularity of this kind of systems is that it involves not only questions and answers but also the user to perform actions and report the results. In other words, the user is a person with the ability to carry out all the required sensory and mechanical operations to solve the problem but without sufficient knowledge while the system has adequate knowledge to solve the problem but cannot perform the required sensory and mechanical operations in order to physically solve it. Collaboration between the user and the machine is therefore necessary.

Tutoring

Nowadays, interactive teaching and tutoring systems are becoming very attractive especially with the emergence of the e-Learning techniques [Rose et al, 1999]. In a general manner, teaching or tutoring is not only providing static information (even if exhaustive) about a topic. A good interactive tutoring system should detect lacks in user's knowledge and provide information consequently [Graesser et al, 2001]. Some systems go a step further by introducing tutoring agents providing online information about the HMD system itself [Hakulinen et al, 2003].

Social Conversation

Some HMD systems do not have another purpose than to converse with users. They are not considered anymore as task-oriented. As already mentioned before, the ELIZA [Weizenbaum, 1966] software is such a system. Several 'ELIZA-like' systems, called chat robots or chat bots, were created since the sixties (in the aim of winning the Loebner Price that goes to AI systems passing the Turing Test) and can be easily found on the Internet. Other works have also been conducted in order to create robots able to take part to group conversations [Matsusaka et al, 2001].

Others

Many other applications can be found to HMD systems. 'Command and control', for example, is one of the tasks that will probably become more popular in the next decade. An operator needing to keep his hands and eyes free could command a machine by voice or gestures. Wearable interfaces are likely to enter in our everyday life rather quickly enabling control of distant devices [Sawhney & Schmandt, 1998]. Anyone will probably want to control TV sets, VCR or DVD players in a more natural, unified and interactive way in the close future.

II.2.3. Levels of communication in a speech-based interaction

During their whole life, human beings learn to produce speech coherently in the purpose of expressing their thoughts. Unconsciously everything around them influences this learning process and it finally becomes totally instinctive.

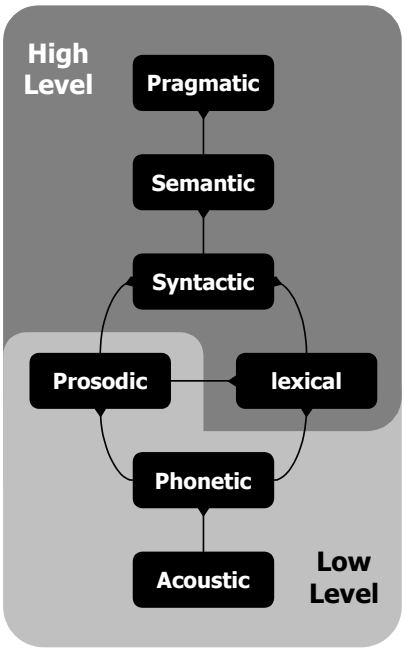


Fig. 1: Levels in a speech-based communication

Formalizing speech-based communication needs the precise description of intuitive mechanisms including both speech production and speech analysis or understanding. It is

a complex introspective process that began to be formally investigated in the mid-fifties while mankind learned to speak for thousand years.

Information conveyed by speech can be analysed at several levels. In the field of NLP, seven levels are commonly admitted in order to describe speech-based communication as depicted on Fig. 1 [Boite et al, 2000]. The figure also shows that levels can be classified into high and low levels of description: the lower the level, the closer it is from the physical sound signal. This distinction between high and low levels is applicable to all types of communications as there is always a possibility to distinguish physical stimuli and their interpretation. Nevertheless, the granularity described in the following is not kept when describing other means of communication.

The Acoustic Level

Speech is firstly a sequence of sounds and so, it is a variation of the air pressure produced by the vocal tract. The acoustic level is the lowest as it concerns exclusively the signal. The study of the acoustic signal includes the study of any of its representation as the electrical output of a microphone (analog or digital), wave forms, frequency analysis (Fourier transforms, spectrograms) etc. Useful information can be obtained from the analysis of the acoustic signal such as the pitch (fundamental frequency), the energy and the spectrum. In general, it is the only level considered by speech coding techniques.

As the human vocal tract is a physical instrument, it is subject to a certain inertia and thus, it cannot assume sudden state modifications. This results in an important property of the acoustic speech signal: it can be considered as a pseudo-stationary signal. Actually, it has been demonstrated that it can even be regarded as stationary along short length time windows of about 30ms.

The Phonetic Level

At this stage, the signal properties are still in focus (low level description) but a step further. Indeed, the phoneticians' focus of interest is the production of particular sounds by the ariculatory system. The physical process of speech production can shortly be described as follow.

The breathing is the primary source of energy. Air is propelled through the vocal tract and different obstacles modulate the air pressure at the output of the mouth and nose. The phonetics studies how humans voluntary contracts muscles in order to dispose obstacles like tongue, lips, teeth and other organs in the aim of pronouncing a specific sound.

The Prosodic Level

The prosodic level, also called more generally the phonologic level by linguists, is the first step toward meanings. Indeed, if the study of phonetics deals with the natural production of different sounds, phonology implies the analysis of a limited number of distinct sounds allowed in a particular language (phonemes), the rhythm with which they are produced in a sequence, the musicality applied to this sequence (prosody) and accentuated part within the sequence. In some languages such as Chinese, changes in the prosody can even completely change the meaning of a word. This level can be considered as transitory between low and high levels as it concerns physically observable features of the signal but those specific traits are voluntary produced by the speaker in the aim of including meaning clues into the speech signal. Prosody is sometimes used to detect

emotion in the speech signal; it is also useful in tutoring applications [Litman & Forbes, 2003].

The Lexical Level

As said before, there is a finite number of different sounds in a specified language: the phonemes. Nevertheless, one cannot utter any sequence of those sounds and produce a valid word according to the particular language. At the lexical level, the focus is on all the valid phoneme sequences that produce words included in the word collection (the *lexicon*) of a particular language. For linguists, this level is sometimes called the morphological level and constitutes also the study of word elementary sub-units that convey sense. This level is the first to be completely associated with the higher levels of communication as it deals with words and thus, can be studied on written language as well without referring to the physical signal.

The Fig. 1, shows that prosody is related to the lexical level like the phonetics. Indeed, in certain languages (like Chinese), different prosodic patterns applied to similar sequences of phonemes produce different words and so, different meanings.

The Syntactic Level

In the same way that all the phoneme sequences are not valid in a given language, legal word chains are constrained by a syntax described in a set of rules: the *grammar*. For one language and its corresponding syntax, there might be several grammars as there are different rule sets suitable to describe the syntax. The grammar also assigns a function to each word in a sentence and so, describes the syntactic structure of the sentence.

In general, grammars written by pure linguists like those taught at school in order to learn languages are not appropriate for computational use. This is why computational grammars have been developed in the early ages of NLU [Chomsky, 1965].

The Semantic Level

Even if an utterance is syntactically correct, there is no assurance that it provides coherent information. Thus, the next step in the description of communication is to ensure that the sentence have sense and to be able to extract that sense from utterances. At the syntactic level starts the study of context-independent meaning, analysing what words mean but also how those meanings combine in sentences.

The Pragmatic Level

Pragmatics groups all the context-dependent information in communication. Indeed, sometimes utterances implicitly refer to underlying information supposed to be known by other participants of a conversation. The underlying information can be the context but also something more general called '*ground knowledge*' that includes all what people from a same culture are supposed to know, their background and their common knowledge. Sometimes, the pragmatic level is divided into three sub-levels [Allen, 1987]:

- Pure pragmatic level: the study of the different meanings that can convey a single sentence uttered in different contexts.

- Discourse Level: concerns how the directly preceding sentence affects the interpretation of the next sentence. It can be helpful for resolving anaphora and other language ambiguities.
- World knowledge level: it is sometimes referred to as 'ground knowledge' as said before and includes all what people know about the world (milk is white for example) but also what an conversation participant knows about other participants beliefs and goals.

II.2.4. General Architecture of a Dialogue System

Human-human dialogues are generally multimodal in that sense that when a person engage a conversation with another, he/she integrates information coming from all his/her senses combined to his/her background knowledge to understand his/her interlocutor. Multiple means are also used to communicate thoughts like gesture, drawings, etc. Sometimes pieces of information coming from different sources are complementary, sometimes they are redundant.

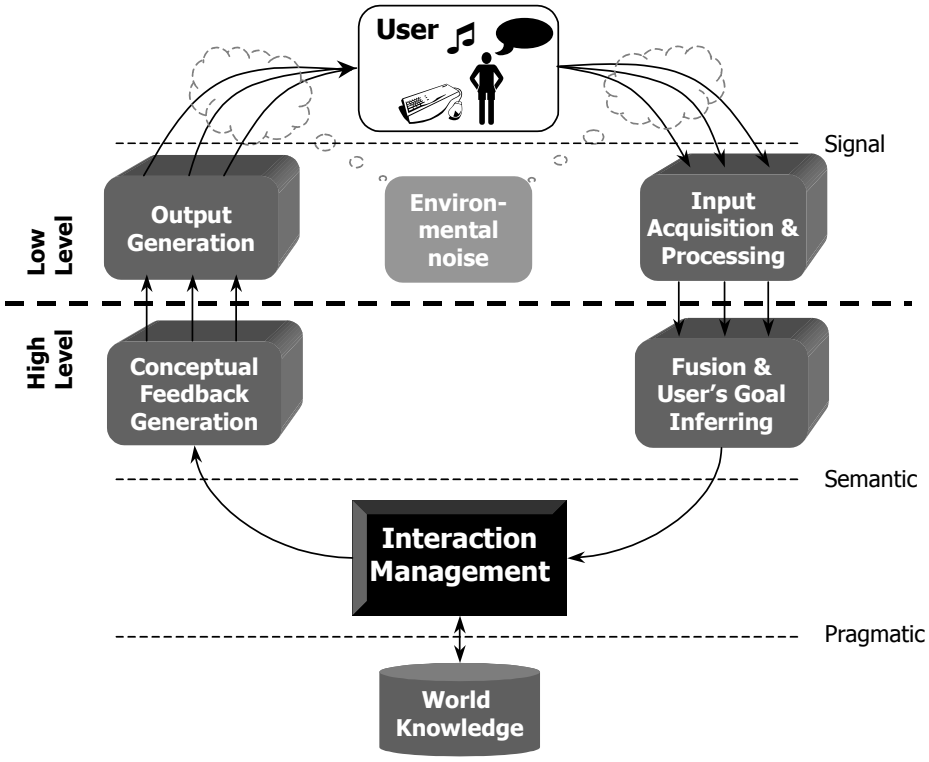


Fig. 2: Typical architecture of a multimodal dialogue system

Naturally, researches in the field of multimodal systems are currently conducted in the purpose of building systems interacting with human users through several means of communication such as gestures [Quek et al, 2002], lips reading [Dupont & Luettin, 2000], humming [Ghias et al, 1995], etc. Multimodality is used either for ergonomics allowing users to introduce complementary information by different means (gesture, humming) or for robustness using redundant information to improve performances (lips reading).

The general architecture of a multimodal dialogue system is depicted at Fig. 2. It shows the typical flow of information going from the user to the system, using any kind of modalities such as gesture, speech, keyboard entry, mouse movements etc. The information is processed and then passed to a sub-system that aims at managing the interaction and has to take decisions about what to do next. According to this decision, new information is transmitted back to the user. Inputs and outputs are processed by a set of sub-systems that will be described more precisely in the following.

Inputs and outputs of such a system are generally perturbed by environmental noise. Noise can be of several natures in function of the type of information it affects and the way this information is conveyed from the user to the acquisition sub-system or from the system back to the user. Of course it can be additive background sounds in the case of audio inputs or convolutive noise in the case of signal transmission over a poor quality channel. Other types of noise can be imagined. For example, in the case of gesture inputs via a video stream captured by a distant camera, any event happening around the user can be mixed up with useful information and is also considered as noise.

Input Acquisition & Processing

As human beings collect information from the outside world using their senses, input signals have to be first captured and transformed by sensors in order to be processed by artificial systems. Sensors can be of different kind such as sets of cameras, microphone arrays, touch pads, joysticks, keyboards, haptic data gloves, etc. These devices usually first convert analogue into digital signals that can be processed by computers or any DSP unit. The sub-system in charge of the acquisition of input signals is generally responsible for their pre-processing as well. Denoising and feature extraction are some steps of the signal pre-processing.

The feature extraction operation aims to reduce the amount of data to be processed by the subsequent sub-systems. Indeed, a full video stream, for example, is not only a to large amount of data to be processed for a dialogue system but also totally inadequate by itself if it must be use for gesture recognition. Actually, only some characteristic points and their moves are useful. The relevant points and information about their moves are so extracted from the video stream and constitutes the features that are passed to the rest of the system.

The denoising process occurs before or after the feature extraction process or even both. In the case of acoustic signals, it is sometimes interesting to apply denoising techniques to the original signal in order to improve the accuracy of extracted features. In the case of gesture recognition, noise introduced by movements around the user can be suppressed by foreground-background segmentation, which is a process applied to both the original signal and to features.

Fusion and User's Goal Inferring

Once proper features are extracted from the original input signals, they are passed to a second sub-system that aims at obtaining meanings (semantic level) and inferring the user's actions and intentions.

As said before, the multi-sensory input data can be redundant or complementary. This means that there might exist dependencies among sets of data and those dependencies may be time-varying. Although some multimodal systems use very primitive data-fusion techniques, the most efficient systems use complex statistical models to infer the user's

goal. Bayesian Networks (BN) (also called Bayesian Belief Networks), Dynamic Bayesian Networks (DBN) [Pearl 1988] and Influence Diagrams [Howard & Matheson, 1984] are general statistical frameworks often used in multimodal systems [Wachsmuth & Sagerer, 2002], [Garg et al, 2000] because of their ability to learn dependencies among sets of input data. Kalman filters and HMMs are particular examples of DBN and recent researches introduced the concept of Asynchronous HMMs [Bengio, 2003] for audio-visual signal processing.

Interaction Manager

The Interaction Manager controls the entire behaviour of the system. After having understood the user's intention, the system has to take a decision about what to do next. This module coordinates the interaction with the user but also with the rest of its environment. Indeed, the system can interact with databases, for example, to retrieve information needed either by the Interaction Manager itself in order to continue its exchange with the user, or by the user. One can imagine that the Interaction Manager could interact with other devices, in order to collect information concerning everything but the user (like external temperature, time of the day, etc.) and that can be helpful to take a decision about the next action to perform. All those possible external devices providing extra information (pragmatic level) are grouped in what was called the 'World Knowledge' on Fig. 2.

The reaction of the system to the user's inputs depends not only on the inferred user's intention but, of course, on the actual task. The Interaction Manager is then the more task-dependent part of the system and is rarely reusable. Challenges of designing an Interaction Manager and particularly in the case of a SDS will be discussed later.

Conceptual Feedback Generation

When the Interaction Manager has taken the decision about the next action, it has to communicate some pieces of information to the user according to this decision. This process starts with the production of a set of concepts supposed to represent this information as well as possible. Indeed, when one is solicited by another human during a dialogue, one first thinks about what has been uttered and then produces a series of ideas or concepts expressing the results of one's thoughts. No words or gestures are yet used but a structure of what one thinks to summarise one's thoughts is built. In the same way, the 'Conceptual Feedback Generation' sub-system builds a set of concepts to be conveyed by any means to the user. This process is often referred to as 'Planning'.

Output Generation

This last sub-system is in charge of using any means at its disposal (speech, screens, sounds, etc.) to express the previously generated concepts in a way that the user can understand them. In the literature, this process is called 'Surface Realization'.

II.3. General Architecture of Spoken Dialogue Systems

Although the learning methods developed in the framework of this thesis are applicable to all types of dialogue systems, this text particularly focuses on systems using speech as the main modality. Those systems are usually referred to as Spoken Dialogue Systems (SDS).

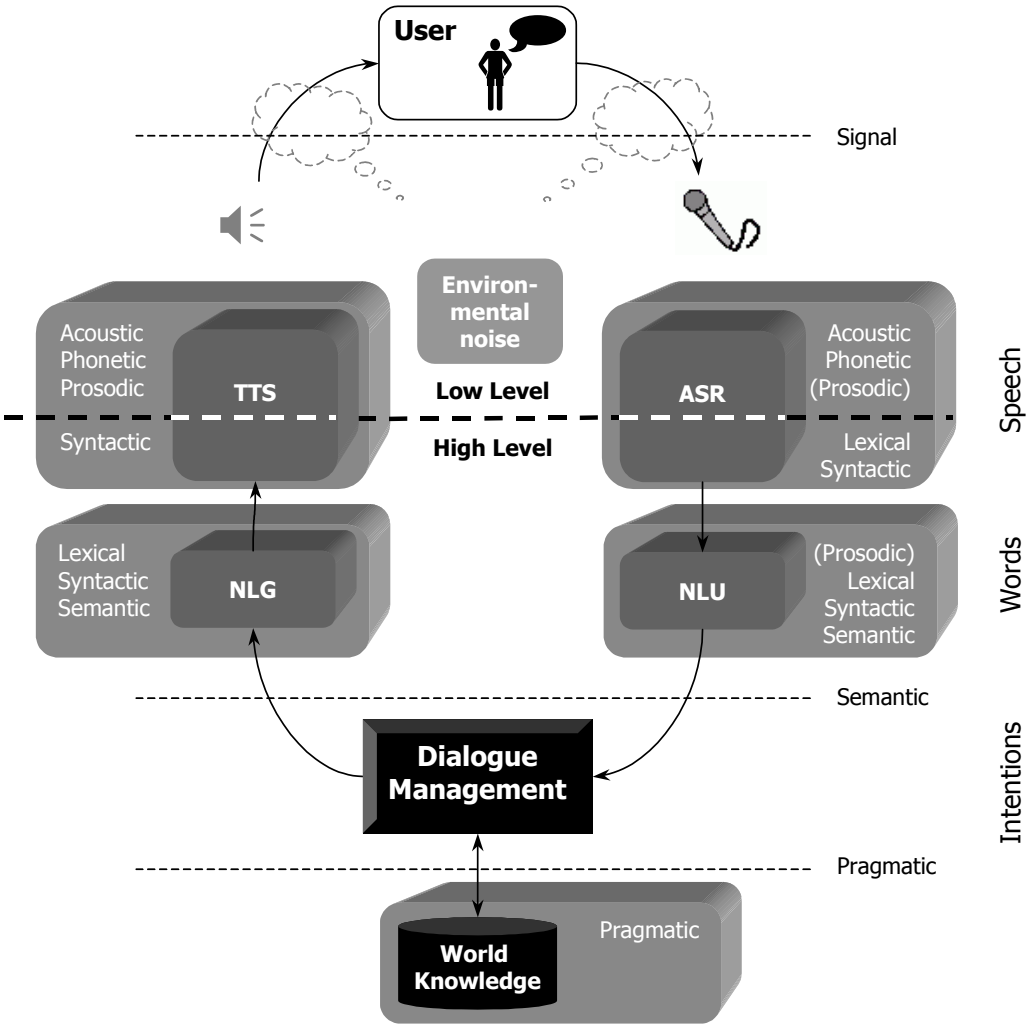


Fig. 3: General Architecture of a SDS

As shown on Fig. 3, a SDS is composed of input processing sub-systems (namely the Automatic Speech Recognition (ASR) and the Natural Language Understanding (NLU) sub-systems), output generation sub-systems (namely the Natural Language Generation (NLG) and the Text-To-Speech (TTS) sub-systems), a Dialogue Management (DM) sub-system and a World Knowledge (WK) base. In this section, the general architecture of sub-systems surrounding the DM will be discussed (as it is the main subject of this thesis, the DM will be the subject of another section) and an overview of their internal functioning will be given as well. Without entering too deeply into details, this section will also show how and why outputs of each sub-system can be prone to errors.

Besides the different levels of speech communication are also depicted on Fig. 3 and associated to the sub-systems working at this level. Yet, three new levels are also defined: the speech, word and intention (or concept) levels. These new abstract levels are very important when considering a SDS since they are higher-level concepts that can be more easily manipulated. They were implicitly defined in section II.2.1.

II.3.1. Automatic Speech Recognition

The Automatic Speech Recognition (ASR) process aims at transcribing into text what has been uttered by the human user. At least, it is supposed to extract data usable by subsequent sub-systems from the acoustic signal in which speech and noise are often mixed up. Thus, in our definition of a SDS (Fig. 3), the ASR sub-system includes the signal acquisition devices. On Fig. 4, the complete ASR process is depicted. On this figure, VAD stands for Voice Activity Detector.

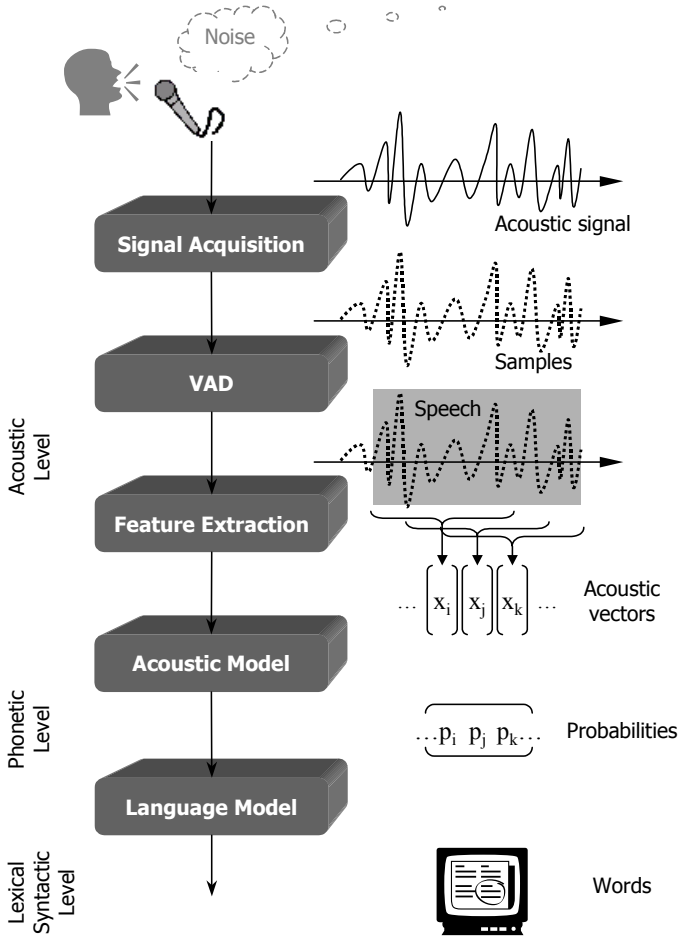


Fig. 4: ASR process

Signal Acquisition

The ASR process being a complete DSP operation, the signal acquisition sub-system transforms the input acoustic signal in a sequence of digital samples suitable for computer-based analysis. To do so, the electrical signal outgoing from the microphone is firstly amplified and then filtered to avoid aliasing. As the spectrum of voice can be considered as limited to a 4000 Hz band, the cutting frequency of the low-pass filter is often set to this upper bound. The resulting analogue signal is then sampled at equally distant discrete time instants and each sampled value is quantified (Fig. 5).

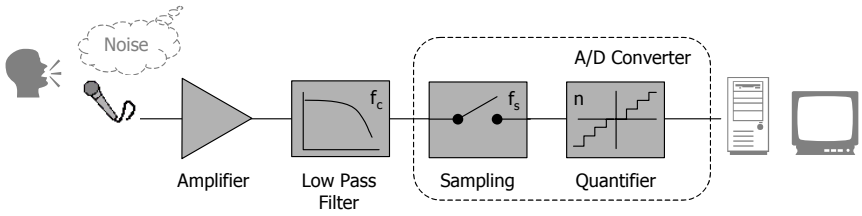


Fig. 5: Signal acquisition process

On Fig. 5, f_c, f_s and n are namely the cutting frequency of the low pass filter, the sampling frequency and the number of bits of the Analogue/Digital Converter. Typical values are:

- $f_s = 2.f_c = 8$ kHz (or $f_s = 16$ kHz)
- n = 16 bits

Voice Activity Detector

The Voice Activity Detector (VAD) aims at distinguishing voice signal from noise or silence. It is not always present but it can be useful for several purposes:

- Avoid useless consuming of computational resources: speech recognition is very cumbersome and it is completely useless to try to recognise anything when only noise or silence is present in the acquired signal.
- Noise modelling: as said in section II.2.4, it is often necessary to apply noise cancellation to the original speech signal in order to improve performance. Lots of denoising algorithms are based on noise modelling. Thanks to the VAD, it is possible to build a model of the environmental noise as it can be identified and measured between voice segments. Thus, the role of the VAD is not only to extract voice segments but also to isolate noise segments and both tasks are as useful.
- Enabling barge-in: sometimes it can be ergonomically interesting to allow users to stop system's prompts by speaking over it. The VAD can do that job.

There exist a lot of voice activity detection algorithms based on the signal's energy, more or less complex spectral analysis, etc but none has really become a standard. This is why no specific method is detailed here.

One particular problem of voice activity detection is that voice is precisely not considered as noise. This means that the VAD cannot make the distinction between useful speech and speech uttered by people surrounding the user.

Feature Extraction

As said before, since the vocal tract is a physical device, it cannot change it state instantaneously. This results in the pseudo-stationarity of the speech signal that can be considered as stationary in a time window of 30ms. If speech is sampled with a frequency of 8 kHz, 240 samples are necessary to represent a 30ms frame. Direct use of raw speech signal samples for speech recognition or any other speech processing is too cumbersome and time-consuming. The sample sequence is instead processed to reduce the data stream and take advantage of speech redundancy. The complete process of feature extraction is shown on Fig. 6.

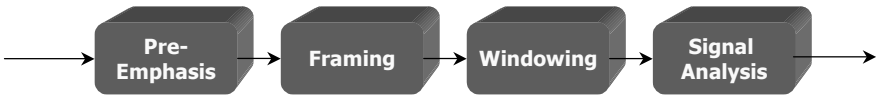


Fig. 6: Feature extraction process

Before any processing, the sampled signal is often submitted to a pre-emphasis that aims at accentuating high frequencies and thus, flattens the signal spectrum. The sampled signal is passed through a filter or the form:

$$\mathrm{H}(z) = 1 - az^{-1} \text{ with } 0.9 < a < 1 \tag{II.1}$$

Then, the framing process divides the sample flow in a sequence of frames each containing the number of samples corresponding to 30ms of speech and computed every 10ms. That is, two consecutive analysis frames overlap each other over 20ms and frames are extracted with a frequency of 100 Hz. Overlapping is necessary to take the context of each frame into account.

Actually, framing corresponds to applying a rectangular window to the original signal. This induces well-known edge effects in local computation like spectral analysis. To attenuate edge effects, samples from each frame are multiplied by a window function, which can be one of the following:

$$\begin{aligned} \mathrm{Hanning}(\mathrm{n}) &= 0.5 + 0.4.cos\left(2\pi.\tfrac{\mathrm{n}}{\mathrm{N-1}}\right) \\ \mathrm{Hamming}(\mathrm{n}) &= 0.54 + 0.46.cos\left(2\pi.\tfrac{\mathrm{n}}{\mathrm{N-1}}\right) \end{aligned} \tag{II.2}$$

Eventually, the signal analysis aims at effectively reducing the amount of data to be processed by extracting one k-size feature vector $x_n = \{x_{n1}, \ldots, x_{nk}\}$ for each N-size sample window. A good feature vector will of course convey the same amount of information than the original sampled signal for purpose of speech recognition by exploiting the signal redundancy. Analysis techniques are generally based on the modelling of the spectral envelope of the speech signal. This envelope presents a number of smooth peaks called formants corresponding to resonance frequency of the vocal tract. There are two main families of signal analysis, respectively the linear predictive (or auto-regressive) and filter bank analysis

The first technique, also called Linear Prediction Coding (LPC) analysis, consists in modelling the speech production process by a simple speech synthesis model composed of an excitation module followed by a filter of the form:

$$H(z) = \frac{1}{1 - \sum_{\mathrm{i-1}}^{\mathrm{k}} a_i z^{-\mathrm{i}}} \tag{II.3}$$

In general 11-order to 13-order filters are used to model each analysis window. The coefficients of the autoregressive filter are not commonly used to directly populate the feature vector but they can be used to compute more popular features like Parcor parameters or the cepstral parameters (or cepstrum) [Rabiner & Juang, 1993].

The filter bank analysis method consists in applying a set of contiguous symmetric filters to the sampled signal and in extracting one coefficient for each filter. The central frequencies of filters can be equally distant or follow a non-linear scale like the Mel or the Bark scale. Those very similar scales are mapping the linear Hertz frequency scale to a perceptual scale deduced from psycho-acoustic experiments. For example, the extraction of Mel Frequency Cepstral Coefficients (MFCC) needs a 20-filter bank disposed on the Mel scale [Rabiner & Juang, 1993] while the Perceptual Linear Prediction analysis (PLP) makes use of a 15 filters equally spaced on the Bark scale [Hermansky, 1990].

Acoustic Model and Language Model

The acoustic model aims at associating a feature vector with its corresponding acoustic unit. Thus, after the signal analysis step, the ASR process can be considered as a pattern-matching problem in which each class is an acoustic unit and that can be solved by statistical methods. Acoustic units can be phonemes (section II.2.3), allophones (phoneme given in a specific context) or even phoneme sub-units.

But the ASR problem cannot be summarized to the problem of matching feature vectors to acoustic units. It actually aims at matching acoustic realisations (input acoustic signal) to complex syntactic structure. Although ASR systems of the 70's used direct matching of feature vector sequences to words or even word sequences using Dynamic Time Warping (DTW) techniques [Myers & Rabiner, 1981], all present systems are based on HMMs. HMMs are statistical pattern-matching tools able to integrate local decisions (taken, in this case, by the acoustic model) over time in order to take more global decisions. Thus, they are used as language model. The way HMMs are implemented determines the way the acoustic model has to be implemented, this is why both systems are described in the same section.

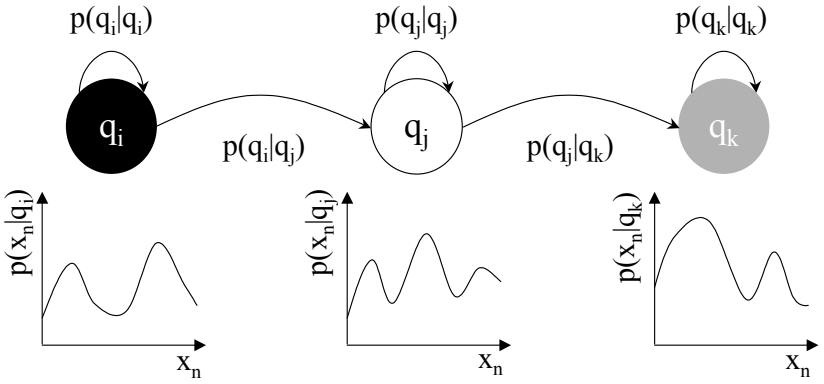


Fig. 7: A 3-state HMM

In the HMM framework, each acoustic unit is modelled by one or several states, words are modelled by sequences of states and sentences are modelled by sequences of words. As shown on Fig. 7, a HMM is fully determined by:

- A set of k states (acoustic classes, typically phonemes): $Q = \{q_1, ..., q_k\}$

- A set of local probability densities describing the emission probability of the feature vector x_n by the class q_i (acoustic model): $p(x_n|q_i)$.
- The one-step dynamic defined by transition probabilities between states: $p(q_i|q_j)$

In this framework, the acoustic model should provide a way to compute $p(x_n|q_i)$ while the dynamic behaviour of the HMM given by $p(q_i|q_j)$ provides integration over time for more global decision. A large model M composed of concatenated elementary HMMs m_i can model complete sentences and the speech recognition problem can be solved by finding the model M that has the highest probability P(M|X) to be the sequence of HMMs that generated the observed sequence of feature vectors $X = \{x_n\}$. Using the Bayes' rule:

$$P(M \mid X) = \frac{P(X \mid M) \cdot P(M)}{P(X)} \tag{II.4}$$

This last equation relates the probability that the acoustic vector sequence X has really been generated by model M (P(M|X)) to:

- the probability that model M generates the acoustic vector X among all the acoustic vectors it can generate: P(X|M) (acoustic model),
- the absolute probability of model M : P(M) (probability of occurrence of the sentence modelled by M in the studied language: language model),
- the *a priori* probability of occurrence of the acoustic vector sequence X: P(X) (supposed to be constant over the maximisation process).

P(X|M) can be computed if all HMMs parameters are known (emission and transition probabilities) thanks, for instance, to a Viterbi algorithm [Viterbi, 1967] [Rabiner & Juang, 1993]. All those probabilities, as well as P(M), are generally estimated by training on data corpus (data-driven technique). By the way, in practice, handcrafted rules are sometimes written to define authorized sequences of words at a given time. The language model is then given by a Finite State Grammar (FSG). Reducing the amount of possible word sequences obviously allows speeding up the recognition process but above all, enhances the performance. Moreover, this permits a first syntactic (or event semantic) pre-processing.

For example, emission probability densities can be estimated by Gaussian Mixture Models (GMM) [Juang *et al*, 1986] like on Fig. 7 or by vector quantification of features vectors [Rabiner & Juang, 1993]. But current most popular ASR systems make use of Artificial Neural Networks (ANN) to classify acoustic vectors and to estimate emission probabilities ($P(x_n|q_i)$) [Bourlard & Morgan, 1994].

Typically, ASR systems are able to provide a set of N possible words or word sequences that can match the speech input with a given probability. Members of the set are generally called the N-Bests and the associated probability is called the Confidence Level (CL). In general, emission probability estimates are often used to provide confidence measures [Williams & Renals 1997], [Mengusoglu & Ris, 2001].

Of course, as the ASR process is a statistical process and assumes that several hypotheses are met, it cannot be considered as reliable at 100 percent. Indeed, one can easily understand that three typical errors can occur: insertion, deletion and substitution of words inside the recognized word sequence.

II.3.2. Natural Language Understanding

After a word sequence has been extracted from the input speech signal, it is the job of the NLU sub-system to extract meanings of it. A good NLU system dedicated to spoken language understanding should take into account that the previous processing sub-systems are error-prone and should particularly focus on potential errors introduced by the ASR process. To do so, some NLU systems are closely coupled to the ASR system, using some of its internal results. Others use several N-Bests in order to find out the one that makes more sense. On another hand, the NLU system can improve speech recognition results and provide contextual confidence measures.

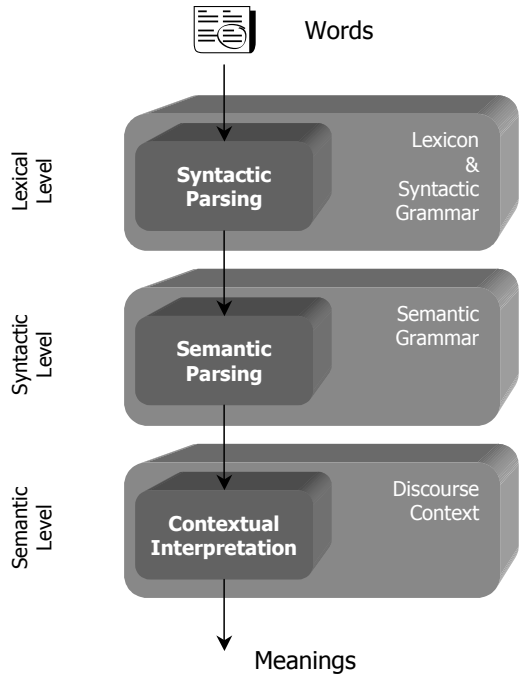


Fig. 8: NLU process

Nevertheless, assuming that the input is a correct word sequence, most of NLU systems can be decomposed as shown on Fig. 8. In the following, basic ideas of NLU will be discussed. For detailed explanations about NLU readers should refer to [Allen, 1987].

Syntactic Parsing

Before trying to extract any sense from a sentence, as a person does, the system should retrieve the syntactic structure of the sentence: the function of each word (part of speech), the way words are related to each other, how they are grouped into phrases and how they can modify each other. Indeed, it is for example important to solve the problem of homographs (homophones) having different possible functions. For instance, the word '*fly*' can be a noun (the insect) or a verb and the word '*flies*' can stand for the plural of the noun or the conjugated verb as shown on Fig. 9. This example shows that ambiguity can sometimes be handled by the syntactic analysis of the sentence.

Most syntactic representations of language are based on the notion of Context-Free Grammars (CFG) [Chomsky, 1956], which represents sentences in terms of what phrases

are subparts of other phrases. Most of syntactic parsing algorithms, aiming at effectively create the mapping between a sentence and its structure, were developed with the goal of analysing programming language rather than natural language [Aho & Ullman, 1972]. Two main techniques for describing grammars and implement parsers are generally used: context-free rewrite rules and transition networks [Woods, 1970].

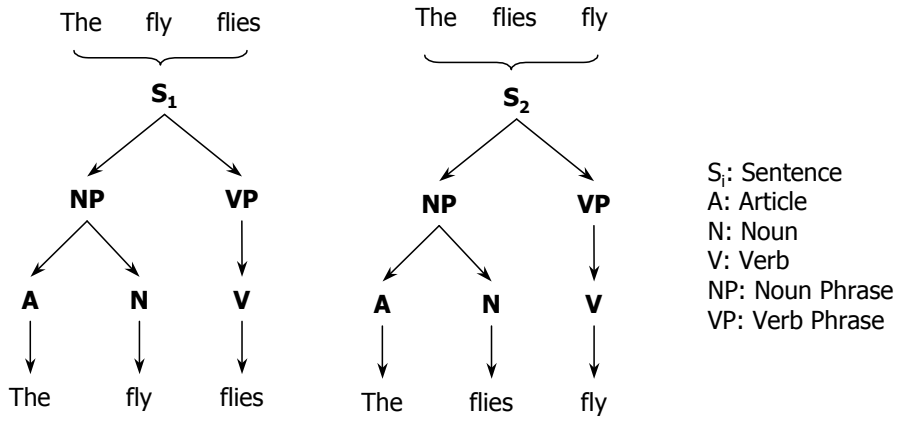


Fig. 9: Syntactic Parsing

For instance, a grammar capturing the syntactic structure of the first sentence in the above example can be expressed by a set of rewrite rules like the following:

1. S → NP VP
2. NP → A N
3. VP → V
4. A → The
5. N → fly
6. V → flies

Rules of the first columns are called 'non-terminal rules' because they can still be decomposed and others are called 'terminal rules' (or terminal symbols). Rewrite rules are very similar to those used in logical programming (like PROLOG). This is why logical programming has been widely used for implementing syntactic parsers [Gazdar & Mellish, 1989].

The above grammar can also be represented by the following State-Transition Network (STN):

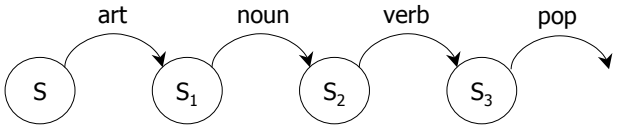


Fig. 10: Transition Network Grammar

In natural languages there are often restrictions between words and phrases. There are many forms of agreements including number agreement, subject-verb agreement, gender agreement, etc. For example, the noun phrase '*a cats*' is not correct in English, as it doesn't satisfy the number agreement restriction. To handle this kind of problems, the previously described formalisms can be extended to allow constituents to have features (like number, gender, etc) and grammars are then called augmented grammars. There are Augmented CFGs and Augmented Transition Networks (ATN).

Although CFGs and ATNs have been the subject of extensive researches, they are limited as they are often handcrafted and deterministic. It is in general difficult to cope with ambiguities when several interpretations are possible. With the emergence of data-driven

and statistical techniques came other solutions to the parsing problem [Jelinek, 1990]. For instance, the part-of-speech tagging problem can be expressed as the problem of selecting the most likely sequence of syntactic categories (C_1, … , C_T) for the words (w_1, … , w_T) in a sentence, that is the sequences (C_1, … , C_T) that maximizes the probability:

$$P(C_1,...,C_T \mid w_1,...,w_T) \tag{II.5}$$

Using the Bayes' rule:

$$P(C_1,...,C_T \mid w_1,...,w_T) = \frac{P(w_1,...,w_T \mid C_1,...,C_T) \cdot P(C_1,...,C_T)}{P(w_1,...,w_T)} \tag{II.6}$$

This last equation doesn't solve the problem as, even with a large amount of data, it is quite impossible to evaluate probabilities of the numerator (as the denominator stays constant, it is ignored for the maximization problem). Two assumptions are generally made.

Firstly, $P(C_1, …, C_T)$ can be approximated by using a n-gram model. That is assuming that the probability of occurrence of category C_i only depends on the n-1 previous categories:

$$P(C_i \mid C_{i-1},...,C_0) \cong P(C_i \mid C_{i-1},...,C_{i-n-1}) \tag{II.7}$$

In particular, a bigram model is often adopted and only $P(C_i \mid C_{i-1})$ is used, which leads to:

$$P(C_1,...,C_T) \cong P(C_1) \cdot \prod_{i=2}^{T} P(C_i \mid C_{i-1}) \tag{II.8}$$

On another hand, it can also be assumed that a word appears in a category independently of the word in the preceding and the succeeding categories:

$$P(w_1,...,w_T \mid C_1,...,C_T) \cong \prod_{i=1}^{T} P(w_i \mid C_i) \tag{II.9}$$

Each $P(w_i \mid C_i)$ can be compared to the emission probability of the ASR problem while each $P(C_i \mid C_{i-1})$ can be compared to a transition probability. The part-of-speech tagging problem is then to find the sequence (C_1, … ,C_T) that maximizes:

$$P(C_1) \cdot P(w_1 \mid C_1) \cdot \prod_{i=2}^{T} P(C_i \mid C_{i-1}) \cdot P(w_i \mid C_i) \tag{II.10}$$

All the probabilities in the above equation can be estimated using a manually tagged data corpus and the problem can then be solved with the same tools used for solving the ASR problem (a Viterby algorithm, for instance).

Semantic Parsing

The semantic parsing aims at extracting the context independent meaning of a sentence. For example, whatever the context, the noun phrase '*video cassette recorder*' has a single meaning and refers to the device that records and plays back video tapes. In the same way, although the word '*flies*' presents an ambiguity, once it has been correctly identified as a verb by the syntactic parser it doesn't need any contextual interpretation to reveal its sense. This shows the utility of the syntactic parsing before any attempt to give a semantic interpretation to a sentence.

Nevertheless, a single word can have several meanings that cannot be disambiguated by a syntactic parsing. Sometimes, synonyms of a word's particular sense can be found, thus a same meaning can also have several realisations. Anyway, in a given sentence, some senses of a word can be eliminated. For instance, in the sentence '*John wears glasses*', the word '*glasses*' means spectacles and not recipients containing something to drink, because John cannot wear recipients. Remaining ambiguities are context-dependent and will be considered in the next sub-section.

From these considerations, it figures out that semantic interpretation is a classification process in which words or groups of words are to be placed in classes regrouping synonyms. The set of classes in a particular representation of the world (or at least of a task) is called its *ontology*. Such classifications have been of interest for a very long time and arise in Aristotle's writings in which he suggested the following major classes: substance, quantity, quality, relation, place, time, position, state, action, affection.

It has also been shown that some information should be enclosed in the semantic representation. For example, the verb is often described as the word in a sentence that expresses the action, while the subject is described as the actor. Other roles can be defined and are formally called *thematic roles*. A good semantic representation should enclose information about thematic roles.

On another hand, utterances are not always used to make simple assertions about the world. For example, the sentence 'Can you pass the salt?' is not a question about the ability of someone to pass the salt but rather a request to actually pass the salt. That means that utterances can have the purpose to give rise to reactions or even to change the state of the world, they are the observable performance of communicative actions. Several actions can be associated to utterances like assertion, request, warning, suggestion, informing, confirmation etc. This is known as the '*Speech Act*' theory and has been widely studied in the field of philosophy [Austin, 1962]. As the Speech Act theory is an attempt to connect language to goals, it is very important to take it into account in the semantic representation of an utterance. Thus, computational implementations of this theory have been early developed [Cohen & Perrault, 1979].

Philosophy provided lots of other contributions to natural language analysis and particularly in semantics (like the lambda calculus, for example) [Montague, 1974].

Given this, Allen proposes to use a particular formalism to map sentences to context independent semantic representations: the *logical form* [Allen, 1987]. Although it is not always used, lots of semantic representation formalisms are closely related to logical forms. Allen also proposes some techniques to go from the syntactic representation to the logical form.

Attempts to apply the previously exposed probabilistic techniques to semantic parsing have been realized [Pieraccini & Levin, 1992]. Indeed, the problem can be expressed as the problem of selecting the most likely sequence of concepts (C_1, … , C_T) for the words (w_1, … , w_T) in a sentence. This problem can be solved assuming the same hypotheses about dependencies of variables along the time and by the same algorithms. It should be noticed that this technique bypasses the syntactic parsing.

Contextual Interpretation

Contextual interpretation takes advantage of the *discourse* level's information to refine the semantic interpretation. The term discourse defines any form of multi-sentence or multi-utterance language. In general, contextual interpretation is used to lift several particular ambiguities staying in the semantic interpretation.

Three main ambiguities can be lifted at the discourse level:

- Anaphors: an anaphora is a part of a sentence that typically replaces a noun phrase. There are several types of anaphora: intrasentence (the reference is in the same sentence), intersentence (the reference is in a previous sentence) and surface anaphors (the reference is evoked but not explicitly referred in any sentence of the discourse).
- Pronouns: pronouns are particular cases of anaphors as they typically refer to noun phrases. Nevertheless, they have been the subjects of lots of specific studies as they are very common.
- Ellipsis: ellipses involve the use of clauses (parts of discourse that can be considered as stand-alone) that are not syntactically complete sentences and often refer to actions or events. For instance, the second clause (B) in the following discourse segment is an ellipsis:

  A. John bought a car.

  B. I did too.

There are several techniques for resolving anaphors but they are almost all based on the concept of Discourse Entity (DE) [Kartunnen, 1976]. A DE can be considered as a possible antecedent for an unresolved ambiguity in the local context and it is typically a possible antecedent for a pronoun. The DE list is a set of constants referring to objects that have been evoked in previous sentences and can subsequently be referred implicitly. Classically, a DE is generated for each noun phrase.

Although there exist simple algorithms for anaphora resolutions based on DE history (the last DE is the most likely to be referred to), the most popular techniques are based on a computational model of discourse's focus [Sidner, 1983] that evolved into the *centering* model [Walker *et al*, 1998 b]. The ideas behind these theories are that the discourse is organized around an object (the *center*), that the discourse is about, and that the center of a sentence is often pronominalised. The basics of the centering theory are based on two interactive structures:

- The preferred next center (C_p): this is the first of the potential next centers' list (or the forward-looking centers: C_f), which are listed in an order reflecting structural preferences (subject, direct object, indirect object…).
- The center, which is what the current sentence is about (C_b). C_b is typically pronominalized.

Generally, three constraints are stated between the center and the anaphora:

1. If any object in the current sentence is referred to by a pronoun, then the center C_b must also be referred to by a pronoun.
2. The center C_b must be the most preferred DE in the current sentence to be referred to by a pronoun.
3. Continuing with the same center from the current sentence to the next is preferred over changing.

Given this, the anaphora resolution is based on the tracking of the center C_b.

The ellipsis resolution is generally based on syntactic constraints. Indeed, the hypothesis underlying most of ellipsis analyses is that the completed structure of the elliptical clause corresponds to the structure of the previous clause. Thus, when a structure matching has

been found between the two clauses, the semantic interpretation of the previous clause can be updated. Remaining ambiguities are solved by semantic closeness [Dalrymple *et al*, 1991].

II.3.3. Natural Language Generation

Natural Language Generation (NLG) systems aim at producing understandable text in a human language, generally starting from a non-linguistic representation of information (concepts). Researches in this field started in the 1970's but reference books are quite recent [Reiter & Dale, 2000]. NLG systems can be used for example to produce documentation about a programming language [Lavoie *et al*, 1997], to summarize e-mails, to provide information about a topic [Goldberg *et al*, 1994], and it starts to be studied in the field of SDS [Rambow *et al*, 2001]. There are lots of NLG techniques and particularly when the generated text should be used for speech synthesis afterward (concept-to-speech synthesis).

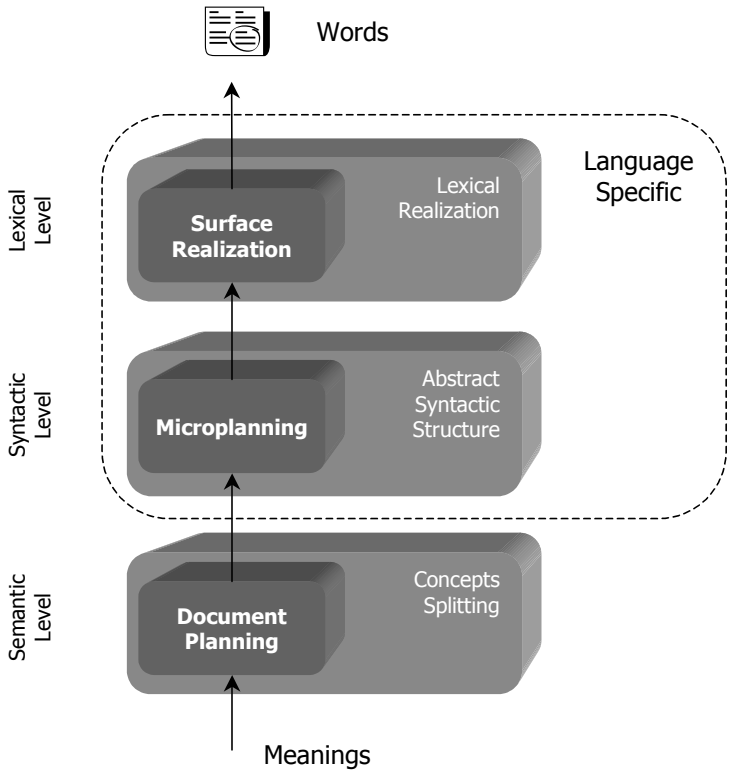


Fig. 11: NLG process

In many SDS systems, the NLG sub-system is very simple and can be one of the following:

- Pre-recorded prompts: sometimes real human voice is still preferred to TTS systems because it is more natural. But this is possible if only a few sentences have to be uttered by the system. In this case, a simple table mapping concepts to corresponding audio records is built. There are obviously lots of drawbacks with this technique as the result is static and recording human speech is often expensive.

- Human authoring: the idea is approximately identical but the table contains written text for each possible concept sequence. The text is subsequently synthesized by a TTS system. This solution is more flexible as written text is easier to produce and modify but still needs human expertise. Nevertheless, this technique is extensively used.

Although using one of those techniques is feasible and widely done in practice, it is not suitable for certain types of dialogue like tutoring or problem-solving for example. In such a case, the system's utterances should be produced in a context-sensitive fashion, for instance by pronominalising anaphoric references, and by using more or less sophisticated phrasing or linguistic style [Walker *et al*, 1996] depending on the state of the dialogue, the expertise of the user, etc. Therefore, more complex NLG methods are used and the process is commonly split into three phases like shown on Fig. 11 [Reiter & Dale, 2000].

Document Planning

NLG can be formalized as a process leading from high-level communicative goals to a sequence of communicative acts (Speech Acts), which accomplish the communicative goals. During the document-planning phase (or text planning phase), high-level communicative goals are broken into structured representations of atomic communicative goals that can be attained with a single communicative act (in language, by uttering a single clause). This phase gives then an overview of the overall structure of the document to be produced and creates an inventory of what should be contained in it. This is very task-dependent and techniques used for document planning are closely related to expert systems techniques. This first phase is considered as language independent while the two following are language dependent.

Microplanning

Microplanning (or sentence planning) is the phase during which abstract linguistic resources are chosen to achieve the communicative goals (lexicalisation). For instance, a specific verb is chosen to express an action etc. Then an abstract syntactic structure for the document is built (Aggregation). This means that microplanning also defines how much clauses will be produced. Finally, in order to produce more natural language, a last process focuses on referring expression generation. Indeed, if several clauses of the generated document refer to the same DE, it can be pronominalized after the first reference. The result is a set of phrase prototypes (or proto-phrases)

Several techniques can be used to perform microplanning. Some are based on 'reversed parsing', that is semantic grammars used to analyse input sentences uttered by the human user are also used to generate document prototypes [Shieber *et al*, 1990]. The idea seems attractive but the development of semantic grammars suitable for both NLG and NLU appeared to be very tricky and in general, same meanings will always be translated into same proto-phrases. Other techniques are based on grammar specifically designed for NLG purposes. Template based techniques are widely used in practice [McRoy *et al*, 2000] and learning techniques have been recently successfully applied to sentence planning in the particular context of SDS [Walker *et al*, 2002].

Surface Realisation

During surface realisation, the abstract structure (proto-phrases) built during microplanning are transformed into surface linguistic utterances by adding function words (such as auxiliaries and determiners), inflecting words, building number agreements, determining word order, etc. The surface realization process even more than the previous one is strongly language dependent and uses resource specific to the target language. This phase is not a planning phase in that it only executes decisions made previously, by using grammatical information about the target language.

II.3.4. Text-To-Speech

The Text-to-Speech (TTS) process aims at converting any text string into an acoustic speech signal. It is very useful in a SDS as it allows uttering any text and thus eases the update and improves the flexibility of the system. It can be used either with a complex NLG system or with human-authored prompts. There is no need to record human prompts (as long a you own a TTS system and you do not develop it) but generally, the subjective quality of synthesized speech is not as good as real voice. As shown on Fig. 12, the TTS process can be split into two main sub-processes: a NLP and a DSP process [Dutoit, 1997].

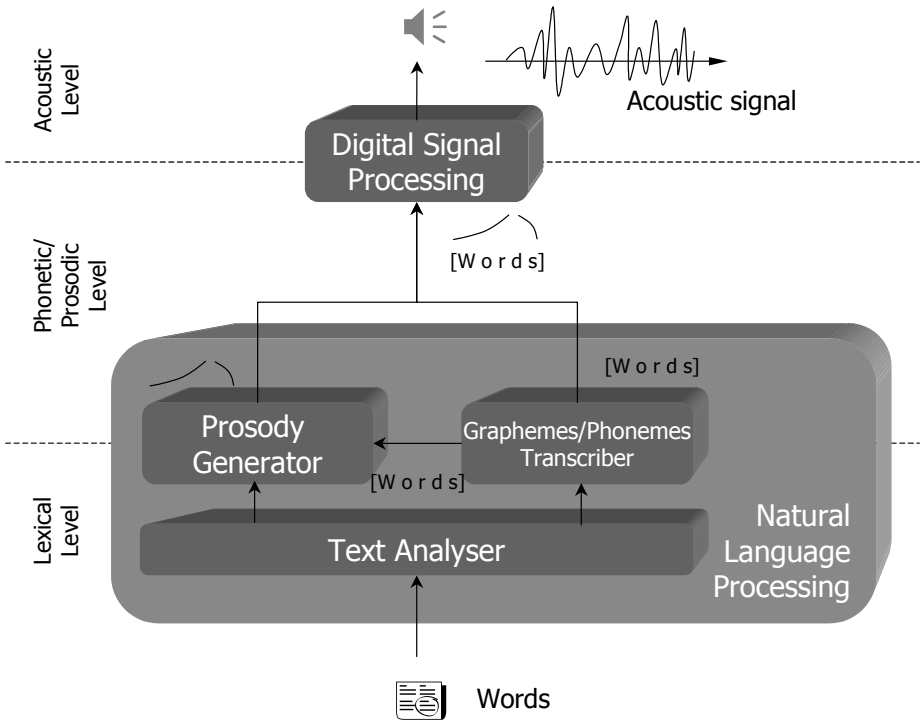


Fig. 12: TTS process

Natural Language Processing

The NLP block pre-processes the text in order to provide suitable information to the DSP block. It is composed of three main blocks as depicted on Fig. 12:

- The *text analyser* is often a syntactic parser that aims at finding function of words in the text to be synthesized. For example, to solve the problem of heterophonic homographs (words that have the same spelling but not the same reading), assigning a part-of-speech to problematic words is often enough to resolve the ambiguity. On another hand, prosody (intonative contour and rhythmic pattern) is often linked to the structure of the sentence: pauses are inserted around punctuation marks… This is why the prosody generator also uses the analysed structure of the sentence. All what have been said so far about parsers remain true for the text analyser.
- The *grapheme-to-phoneme transcriber* provides a phonetic transcription (similar to those provided by dictionaries and shorters) of the input text. This process is extremely language dependent (each language has its own set of phonemes). As one can easily imagine, a simple rule assigning a phoneme to a letter will not perform correctly. Indeed, sometimes one letter is associated to several phonemes (e.g. 'c', 'k', 'q'), groups of letters produce only one phoneme like in diphthongs (e.g. 'an') or a same letter produces different phonemes in different contexts (e.g. 'ce' ≠ 'ca'). There are lots of other problems to solve and techniques usually make use of high-level information such as syntactic and morphological information (provided by the text analyser) [Allen *et al,* 1987]. Rules can also be derived from corpus data [Daelemans & Van den Bosch, 1993].
- Finally, the *prosody generator* has to build an intonative contour that will be applied to the speech output in order to enhance the naturalness of the synthesized acoustic signal. Although early prosodic generation system were rule-based [Crystal, 1969], most of nowadays systems rely on the syntactic analysis of the text. Indeed, it can be shown that a human being is able to apply very convincing intonation on a syntactically correct sentence, no matter if it is semantically correct. Nevertheless, semantic analysis of sentences for prosody purposes can dramatically improve the naturalness of the synthesized speech and particularly when considering contexts larger than the sentence [Davis & Hirschberg, 1988]. In the subsequent block, prosody will be added by modifying essentially the fundamental frequency of the speech signal, which is commonly called the pitch or F_0.

Digital Signal Processing

This part of the system converts a phonetic/prosodic description of a text into an actual speech signal. This process can be compared to the phonatory process consisting in contracting muscles in order to dispose obstacles for the air like tongue, lips, teeth and other organs and to control the vibratory frequency of vocal folds in the aim of pronouncing a specific sound.

Early TTS systems (like the VODER discussed earlier) were parametric synthesizers and evolved into rule-based synthesizers [Allen *et al*, 1987]. In those systems, the desired speech signal is represented as a sequence of dynamic parameters. As explained earlier (see II.3.1) the speech signal can be segmented into slices in which the signal can be considered as stationary. The spectral envelope of each slice presents smooth peaks called

formants. The idea of parametric synthesis is to build filters that produce a similar spectral shape when excited with appropriate signals. Several parametric techniques exist like the LPC technique and the formant technique. In the first technique, the parameters for each slice are the coefficients of the LPC filter and the excitation. The excitation can be periodic for voiced sounds (the period is a parameter) and a white noise for unvoiced sounds. For the formant synthesizers, a cascade of filters produces the desired spectral envelope. Each filter is a second order resonator which parameters are its central frequency and its bandwidth. The excitation is still a parameter of course. Parameters are typically learned from a corpus, but the data are useless after parameters have been derived.

Nowadays, most of high-quality TTS systems are concatenation synthesizers and they are more and more data-driven. The idea that underlies those techniques is that the less DSP is used, the better will be the subjective quality of speech (its naturalness). In this purpose, a more or less large database containing samples of speech units (or segments) is built. Segments are chosen in order to minimize later concatenation problems, this is why diphones, triphones and half-syllables are preferred. Indeed, they include most of the transition and co-articulation features that are the most difficult to produce from scratch. According to the amount of available speech segments, several concatenation policies are possible. If only one sample of each segment is available [Dutoit, 1993], the database is smaller and unit selection is quite simple but the DSP will be more complex in order to produce the target prosody results etc. On another hand, if several samples of each segment are available, the optimal sequence of units is harder to compute but the subsequent DSP will be simpler and the naturalness will be improved. The optimal sequence of segments is chosen as a trade-off between a target cost (estimating the difference between a database unit and the target), and a concatenation cost (estimating the quality of a join between two consecutive units) [Hunt & Black, 1996]. To realise this complex search, the database is often formalized as a state-transition network and a Viterbi algorithm is (once again) used to find the optimal unit sequence according to the target.

II.3.5. World Knowledge

All the sub-systems discussed so far could be stand-alone systems as they are only related to the task by some parameters like speech grammars (FSG) for the ASR sub-system or ontology for the NLU sub-system.

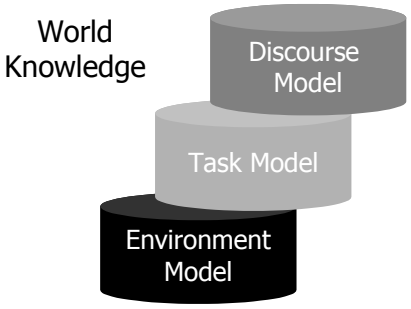


Fig. 13: World Knowledge

However, most of SDSs are task-oriented and thus, the task has to be known in some way by the system in order to provide useful information to the user. By the way, the system

must have certain knowledge about its environment and the history of the interaction in order to perform optimally. Indeed, some information can be retrieved by other means than direct (and costly) interaction with the human user. For those reasons, the World Knowledge (WK) is split as shown on Fig. 13.

Task Model

The task model is a formal representation of the task that is suitable for subsequent processing and reasoning. Several frameworks for describing the tasks have been proposed since the last thirty years and are generally closely related to the type of dialogue management chosen when developing the SDS. The field of AI that studies formal description of knowledge is called Knowledge Representation (KR) and is still in constant development.

A first family of KR is based on the concept of *frame*, introduced in [Minsky, 1975]. It is a relatively complex KR also used in NLU. According to Minsky, the theory is based on the following concept: '*When one encounters a new situation (or makes a substantial change in one's view of the present problem) one selects from memory a structure called a Frame. This is a remembered framework to be adapted to fit reality by changing details as necessary*.' A frame is then described as a data-structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed. Most of the concepts developed in this theory have been extended to even more complex KR like described in [Brachman & Levesque, 1985].

As their problem-solving SDS is based on the *Missing Axiom* and *Theorem Proving* theories, Smith and Hipp built their KR base like a list of axioms helping to achieve the task [Smith & Hipp, 1994]. Each axiom is a rule expressing known information and the dialogue system uses interactions with the user to retrieve axioms that are missing in order to prove the theorem 'the task is achieved'. The whole interaction is then conducted like a demonstration relying on the rules and logic programming is the base of the SDS.

The task can also be represented as an Attribute Value Matrix (AVM) [Walker *et al*, 1997a]. This representation consists in the information that must be exchanged between the SDS and the human user during the dialogue represented as a set of ordered pairs of attributes and their possible values. This KR can be extended to handle multiple right answers (multiple correct values for a same attribute during a dialogue session) and one of the main advantages of this representation is that the information can be organized very systematically into a relational database.

Besides a formal representation of the task, the Task Model should also provide mechanisms allowing transforming outputs of the preceding blocks to include them in the task context. Indeed, after a semantic interpretation have been extracted from the spoken utterance by the NLU sub-system, that doesn't mean that the DM is able to process it in order to produce suitable action. Indeed, there should be a way to translate the sequence of concepts provided by the NLU sub-system into some state representation of the DM and to infer user's *goals* from the concept sequence. Several techniques were proposed for goal inferring. Some techniques are based on heuristics to identify Speech Acts or Communication Acts [Allen, 1987] but more recent systems make use of BNs [Meng *et al*, 2000] or even DBNs [Horvitz *et al*, 1998]. Wathever heuristics, BN or other means are used to infer user's goal, the Task Model part of the WK should contain parameters of the model used.

Discourse Model

First of all, as it has been said before, the resolution of some discourse ambiguities necessitates some pragmatic knowledge [Allen, 1987]. This part of the WK should then include that pragmatic information in order to resolve remaining ambiguities.

Another issue in dialogue management is the keeping and the update of shared *beliefs* along the dialogue session. Indeed, during a dialogue, contributors establish and maintain their mutual belief that their utterances have been well understood. This process has been referred to as *grounding* [Clark & Shaefer, 1989]. In order to provide information, human contributors typically do more than just utter the right sentence at the right time. They coordinate the presentation and acceptance of their utterances until they have reached a sufficient level of mutual understanding to move on and provide new information. An utterance is considered as instructive if it provides information that was not in the history of beliefs. A formal theory of grounding has been recently used in widely distributed software [Paek & Horvitz, 2000]. It is also the role of the Discourse Model to serve as a memory of beliefs and their degree of certainty.

Environment Model

The WK should also keep parameters allowing modelling the environment of the SDS, or more precisely of the Dialogue Manager (DM). Indeed, the SDS should behave according to the performance of all the sub-systems that compose it. The Environment Model should then contain parameters allowing foreseeing ASR, NLU, NLG or TTS problems in order to avoid it [Litman *et al*, 1999].

In the same way, the SDS should not behave the same when interacting with users with different levels of knowledge and expertise about the task and also about the system itself [Veldhuijzen van Zanten, 1999]. Several studies have shown that users adapt themselves to the system and the task as well [Kamm *et al*, 1998]. Thus, even with a same user, the system should adapt its behaviour.

Finally, external events can occur and modify the state of the system. New information can be provided by peripherals like measure instruments, databases, market fluctuation [Meng *et al*, 2001] etc. The Environment Model is independent from the task but can modify the state and the behaviour of the system.

II.4. Dialogue Manager

Introduction

The Dialogue Manager (DM) can be considered as the brain of the SDS. According to the outputs of the early blocks, the history of the dialogue session, information it can find in the WK and its internal *strategy* (mapping of states to actions), the DM takes decisions about what to do next. The main topics developed in this thesis are all closely related to the design of an optimal DM, which means an optimal strategy. However, the definition of optimality in the case of dialogue strategies is not straightforward and until now, no definitive definition has been found despite the amount of recent studies about SDS evaluation.

Moreover, there are several issues in the design of a DM like the depth of the history to be considered (closely related to the way the overall dialogue is thought about), the internal state representation, the possible actions the DM can perform, the level of initiative let by

the system to the user, its adaptability to the environment and to the user, the confidence it can have in the results obtained by the preceding sub-systems and the confirmation sub-dialogues it can enter to improve this confidence etc.

This section, gives an overview of different topics and challenges the design of an optimal DM addresses.

II.4.1. Dialogue Modelling

Before investigating the possibility of building artificial SDS, studies have been conducted in the field of human-human dialogues. One main purpose was to develop a theory of dialogue, including, at least, a theory of cooperative task-oriented dialogue, in which the participants are communicating in the aim of achieving some goal-directed task. Although researchers do not agree on the fact that human-human dialogue should serve as a model for human-machine dialogues (because people adapt their behaviour when talking to machines [Jönsson & Dahlbäck, 1988], [Dahlbäck & Jönsson, 1992]), four approaches to modelling HMD coming from studies about human-human dialogues have been applied in the field of SDS design: dialogue grammars, plan-based models, conversational game models and joint action models.

Dialogue Grammars

Dialogue grammars are an approach with a relatively long history. When adopting this model, the designer of a SDS considers that there exist a number of sequencing regularities in dialogue and that some Speech Acts can only follow some others [Reichman, 1981]. For example, a question is generally followed by an answer (both of them corresponding to different Speech Acts). In the vein of the NLU process, a set of rules state sequential and hierarchical constraints on acceptable dialogues, just as syntactic grammar rules state constraints on grammatically acceptable utterances. Notice that a particular (but very similar) Speech Acts theory has been developed for the purpose of SDS design: the Dialogue Acts theory.

The biggest drawback of dialogue grammars (and what make them so simple) is that they consider that the relevant history for the current state is only the previous sentence (the previous Speech Act or Dialogue Act). This is probably why so few systems were successfully developed based on this model [Dahlbäck & Jönsson, 1992].

Plan-Based Model

Plan-based models are founded on the observation that humans do not just perform actions randomly, but rather they plan their actions to achieve various goals, and in the case of communicative actions (Speech Acts), those goals include changes to the mental state of listeners. Plan-based models of dialogue assume that the speaker's Speech Acts are part of a plan, and the listener's job is to uncover and respond appropriately to the underlying plan, rather than just to the previous utterance [Carberry, 1990]. In other words, when inferring goals from a sentence, the system should take the whole dialogue into account and interpret the sentence in the context of the plan. Those models are more powerful than dialogue grammars but goal inference and decision making in the context of plan-based dialogue are sometimes very complex [Bylander, 1991]. Nevertheless, plan-based dialogue models are used in SDSs [Freedman, 2000].

Conversational Game Theory

The Conversational Game Theory (CGT) [Power, 1979] is an attempt to include the ideas from both plan-based models and dialogue grammars in the same framework. From this point of view, dialogues consist of exchanges called games. Each game is composed with a sequence of moves that are valid according to a set of rules (similar to grammars) and the overall game has a goal planned by participant agents (similar to plan-based models). Thus, agents share knowledge (beliefs and goals) during the dialogue and games can be nested (sub-dialogues are possible) to achieve sub-goals. In this framework, moves are often assimilated to Speech Acts [Houghton & Isard, 1987]. The CGT defines quite formally the moves allowed for each of the participants at a given instant in the game (according to the rules and the goal) and thus, allows simulation of dialogues too [Power, 1979]. Computational versions of CGT have also been developed and implemented in working SDSs [Lewin, 2000].

Joint Action Model

The previous approaches considered a dialogue as a product of the interaction of a plan generator (the user) and a plan recogniser (the SDS) working in harmony, but it does not explain why participants ask clarification questions, why they confirm etc. Another dialogue model is emerging in which dialogue is regarded as a joint activity, something that agents do together. Participating in a dialogue requires the participants to have at least a joint agreement to understand each other, and this motivates the clarifications and confirmations so frequent in dialogues. This family of dialogue models are of a great interest and have been used in some systems [Edmonds 1993].

II.4.2. Dialogue Management

Although any of the dialogue models described earlier can be used in SDSs in order to interpret semantics, build internal state, etc, the dialogue management can be of several kinds. This section will describe some of possible dialogue management methods found in the literature.

Theorem Proving

This approach to dialogue management has been developed in [Smith & Hipp, 1994] in the framework of a problem-solving SDS aiming at some equipment fixing. The idea underlying this method is that the system tries to demonstrate that the problem is solved (theorem). As in mathematical demonstration, there are several steps to follow and at each step, the system can use axioms (things known for sure) or deduction to move on. If in a given step, the WK is not able to provide an axiom to the DM allowing it to go on or if the DM can not deduce it from other axioms, the axiom is considered as missing (*missing axiom* theory) and the system prompts the user for more information. If the user is not able to provide information, a new theorem has to be solved: 'the user is able to provide relevant information'. A tutoring sub-dialogue then starts. This technique has been implemented in Prolog as this rule-based method is suitable for Logic Programming. The major drawback of theorem proving management is that the strategy is fixed as the demonstration steps are written in advance.

Finite-State Machine

In the case of finite-state methods, the dialogue is represented like a state-transition network where transitions between dialogue states specifie all legal paths through the network. In each state an action is chosen according to the dialogue strategy and the result of the action leads to a new transition between states.

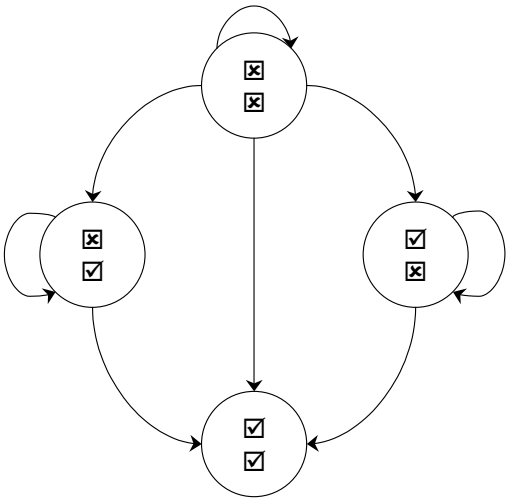


Fig. 14: A four-state SDS

For example, let's imagine a simple SDS that aims at retrieving the first name and last name of the user. Four states are possible like shown on Fig. 14:

- None of the values is known.
- The first name is known and the last name is unknown.
- The last name is known and the first name is unknown.
- Both values are known.

On the figure, each circle represents a possible state and possible transitions between states are represented by directed edges. In each circle, there are two squares representing Boolean values indicating whether or not a value is known (the first name for the upper square and the last name for the lower square). The sign ✗ means that the value is unknown and the sign ✓ means that the value is known.

The starting state is the one where none of the values is known (the upper state) and the goal is to reach the state where both values are known (the lower state). In each state the DM has the choice between several actions like prompting for both values, for the first name or for the last name or closing the dialogue.

The main drawback of these methods is that all possible dialogues have to be known and described and the structure is quite fixed resulting in inflexible and often system-led behaviours. Nevertheless, state-transition methods have been widely used in dialogue systems but above all in various toolkits and authoring environments [McTear, 1998] for several reasons.

- It is an easy manner for modelling dialogues that concern well-structured tasks that can be mapped directly on to a dialogue structure.
- Finite-state method is easier to understand and more intuitive for the designer as a visual, global and ergonomic representation of other

management techniques would be difficult to realize. In short, it is more 'user-friendly'.

- It is much more straightforward to give a visual representation of finite-state as a dialogue is described as a set of states linked by transitions.
- Only user-led systems can't be described by a finite-state model. Most of dialogues are generally system-led or mixed-initiative.
- Scripting languages can be very easily used to represent state-transition networks.
- As discussed in [McTear, 1998], lots of applications like form-filling, database querying or directory interrogation are more successfully processed this way. Those applications are the most popular.
- System-led behaviour is very easy to describe as a state-transition network.
- Even if other methods (like self-organized methods) are definitively more compliant, nested sub-dialogues may help to gain in flexibility in the finite-state methods.

Form Filling

Form filling methods, also called frame-driven, frame-based or slot-based methods, are mainly used when the application consists in a transfer of information from the user to the SDS and when the information can be represented as a set of attribute-value pairs. The attribute-value structure can be seen as a form and the user should provide values for each field (attribute, slot, frame) of the form. Each empty field is then eligible for a prompt from the system to the user. The dialogue strategy aims at filling completely the form and to retrieve values for all the fields. Each field is given a priority defining the sequence in which the user is prompted [Goddeau *et al*, 1996]. Each of the user's utterance has then to be processed in order to provide an attribute-value representation of its meaning.

Self-Organized

Unlike previous dialogue management techniques, the self-organized management family does not require all the dialogues paths to be specified in advance. Each action of the system and reaction of the user contributes to build a new configuration to which is associated a particular behaviour. There is no need to know how the configuration occurred. This is generally referred to as event-driven dialogue management. One of the more famous attempts to use this kind of dialogue management was the Philips Speechmania software based on the HDDL language (Harald's Dialogue Description Language, from the first name of its creator) [Aust & Shroer, 1998]. Other attempts were made to use event-driven dialogue management [Baekgaard, 1996] but the complexity of development of such systems overcame their potentialities.

II.4.3. Degrees of Initiative

One of the main issues in designing a dialogue strategy is the degree of initiative let to the user in each dialogue state. Indeed, it seems obvious that the dialogue management will be easier if the system controls completely the dialogue flow while a user will be more satisfied if he can be over-informative and take initiatives. Three degrees of initiative can be distinguished, each with their advantages and drawbacks:

- **System-Led**: the system has the only initiative and asks sequences of precise questions to the user. The user is then supposed to answer those questions and provide only the information he/she has been asked for.
- **User-Led**: the user has the only initiative and asks for information to the system. The system is supposed to interpret correctly the user's query and to answer those precise questions without asking for more details.
- **Mixed Initiative**: the user and the system share the control to cooperate in order to achieve the user's goal. The user can be over-informative and provide information he has not yet been asked for or even ask the system to perform particular actions while the system can take the control at certain dialogue states in order not to deviate from the correct path leading to goal achievement. The system can also take the control because the performance of the previous systems in the chain is likely to get poorer (because of noise or whatever).

Instinctively, one can state that mixed-initiative systems should perform better from the user point of view. Nevertheless, some researches have shown that users sometimes prefer system-led SDSs because the goal achievement rate is greater [Potjer *et al*, 1996]. Other studies have even shown that, besides the fact that system-led gives better performance with inexperienced users, the mixed-initiative version of the same system does not perform better (objectively neither subjectively) with more experienced users [Walker *et al*, 1997b]. Indeed, those reported researches emphases on the fact that human users adapt their behaviour because they know they are interacting with a machine. They usually accept some constraints in order to obtain a better goal completion rate.

II.4.4. Confirmation Strategies

Another issue in the design of a dialogue strategy is to handle correctly possible errors made by input processing sub-systems. Indeed, it is often very useful to enter in some confirmation or clarification sub-dialogues in the aim of enhancing the system's confidence in its beliefs. Notice that such sub-dialogues also occur in human-human dialogues, because of misunderstandings of all kinds (speech signal as well as meaning).

Nevertheless, the choice of a confirmation strategy is not trivial. When and how the system should engage in a confirmation sub-dialogue are two important questions the dialogue designer should ask himself. As for previous problems, there is no straightforward answer to those questions and different researches led to different conclusions.

First of all, the designer should decide when to engage in a confirmation sub-dialogue. Some system designers argue that it is to the user to detect interpretation problems and that the retrieved information should always be played back for confirmation [McInnes *et al*, 1999]. On another hand, objective measures of the confidence the system can have in the information retrieved from the user can help to make a decision. Some systems use the ASR confidence level [Williams & Renals 1997] as unique information to take a decision about whether or not the information should be confirmed. Then, the question is to know under which level the information should be considered as unsure. Moreover, as a complete SDS owns a NLU sub-system, some confidence in the meanings and context should also enter into account in order to improve the overall confidence and to decide when to ask for confirmation [Komatani & Kawahara, 2000]. Finally, another strategy would be to avoid problematic dialogues by predicting them [Litman *et al*, 1999].

If the need for confirmation is detected, the DM still has the choice between two possible confirmation methods:

- Explicit confirmation: the user is asked explicitly if the recognised information has been correctly understood.
- Implicit confirmation: the information to be confirmed is combined with the request for the next piece of information. This strategy relies on the user instinctively contradicting any incorrect information.

Explicit confirmation has proven to be the most reliable. However, this reliability tends to be at the expense of the naturalness and efficiency of the interaction as it increases the dialogue's length. On another hand, implicit confirmation can cause confusions and has been found to be less reliable in guiding users to successful achievement of their goal. Yet, when information is correctly recognised, it has a significant advantage in terms of the speed and therefore usability of the interaction. The trade-off between reliability and speed has resulted in several attempts to combine the use of both. In general, the choice between one of both confirmation strategies still relies on the confidence level of the information. Most of systems apply implicit confirmation when the system has a quite good confidence in the information while explicit confirmation is chosen when poor confidence level is computed [Bouwman *et al*, 1999].

Eventually, the effectiveness of explicit confirmations has also been the subject of different studies. It appeared that they should have question intonation rather than statement intonation and that the details to be confirmed should be placed near the end of the confirmation message: they should not be followed by a question such as 'Is that correct?' Since users will often speak before or during such a question [McInnes *et al*, 1999].

II.4.5. Evaluation

Introduction

Previous sections underlined the main problems of DM design like the choices of a dialogue model, the degree of initiative the system should leave to the users or the confirmation strategy to deploy. It has also been shown that several studies lead to controversial conclusions for each of those problems. Consequently, it would be of great interest to have a framework for SDS performance evaluation and it is the object of a large field of nowadays researches.

Despite the amount of researches dedicated to the problem of SDS performance evaluation, there is no clear and objective method to solve it. Indeed, performance evaluation of such high-level communicative systems highly relies on the opinion of end-users about their interaction and is therefore strongly subjective. Thus, studies on subjective appreciation of SDSs through user satisfaction surveys have often (and early) been conducted [Polifroni *et al*, 1992]. But even those surveys proved to have non-trivial interpretation. For example, experiments reported in [Devillers & Bonneau-Maynard, 1998] demonstrate as a side-conclusion that the users' appreciation of different strategies depends on the order in which SDSs implementing those strategies were presented for evaluation.

Nevertheless, there have been several attempts to determine the overall system's performance thanks to objective measures made on the SDS components, such as speech recognizer performance with one of the first tries in [Hirschman *et al*, 1990]. Other objective measures taking into account the whole system's behaviour (like the average number of turns per transaction, the task success rate etc.) have been exercised in the aim of evaluating different versions (strategies) of the same SDS [Danieli & Gerbino, 1995] with one of the first tries applied in the SUNDIAL project (and after within the EAGLES framework) [Simpson & Fraser, 1993]. More complex paradigms have been developed afterwards.

PARADISE

One of the most popular frameworks for SDS evaluation is the PARADISE (PARAdigm for DIalogue Systems Evaluation) paradigm [Walker *et al*, 1997a]. PARADISE is the first attempt to explain users' satisfaction as a linear combination of objective measures. For the purpose of evaluation, the task is described as an AVM and the user's satisfaction as the combination of a task completion measure (κ) and a dialogue cost expressed as a weighted sum of objective measures (c_i). The overall system's performance is then approximated by:

$$P(U) = \alpha \cdot N(\kappa) - \sum_i w_i \cdot N(c_i), \tag{I.11}$$

where N is a Z-score normalization function that normalises the results to have mean 0 and standard deviation 1. This way, each weight (α and w_i) will express the relative importance of each term of the sum in the performance of the system.

The task completion measure κ is the Kappa coefficient [Carletta, 1996] that is computed from a confusion matrix M summarizing how well the transfer of information performed between the user and the system when using the SDS to be evaluated. M is a square matrix of dimension n (number of values in the AVM) where each m_{ij} is the number of dialogues in which the value i was interpreted while value j was meant. The kappa coefficient is then computed by:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}, \tag{I.12}$$

where P(A) is the proportion of correct interpretations (sum of the diagonal elements of M: m_{ii}) and P(E) is the proportion of correct interpretations occurring by chance. One can see that $\kappa = 1$ when the system performs perfect interpretation (P(A) = 1) and $\kappa = 0$ when the only correct interpretations were obtained by chance (P(A) = P(E)).

In order to compute weights α and w_i, a large number of users are asked to answer a satisfaction survey after having used the system while costs c_i are measured during the

interaction. The questionnaire comprises around 9 statements on a five-point Likert scale and the overall satisfaction is computed as the mean value of collected ratings. A Multivariate Linear Regression (MLR) is then applied with the result of the survey as the dependent variable and the weights as independent variables.

Several criticisms can be made about assumptions and methods used in the PARADISE framework. First, the assumption of independency of the different costs c_i made when building an additive evaluation function has never been proved to be true (it is actually false as the number of turns and the time duration of a dialogue session are heavily correlated for example [Larsen, 2003]). The Kappa coefficient as a measure of task success can also be discussed, as it is often very difficult to compute when too many values are possible for a given attribute. An example is given when trying to apply PARADISE to the PADIS system (Philips Automatic Directory Information System) [Bouwman & Hulstijn, 1998]. Recent studies have also criticised the satisfaction questionnaire. While [Sneele & Waals, 2003] proposes to add a single statement rating the overall performance of the system on a 10-point scale, [Larsen, 2003] recommends to rebuild the whole questionnaire, taking psychometric factors into account (which seems to be a good idea). Finally, the AVM representation of the task has proved to be very difficult to extend to multimodal systems and thus, seems not to be optimal for system comparisons. Some attempts to modify PARADISE have been proposed [Beringer *et al*, 2002].

Besides the abovementioned critiques, PARADISE has been applied on a wide range of systems. It was adopted as the evaluation framework for the DARPA Communicator project and applied to the official 2000 and 2001 evaluation experiments [Walker *et al*, 2000]. Nevertheless, experiments on different SDSs reached different conclusions. PARADISE's developers themselves found contradicting results and reported that time duration was weakly correlated with user satisfaction in [Walker *et al*, 1999] while [Walker *et al*, 2001] reports that dialogue duration, task success and ASR performance were good predictors of user's satisfaction. On another hand, [Rahim *et al*, 2001] reports a negative correlation between user's satisfaction and dialogue duration because users hung up when unsatisfied. Finally [Larsen, 1999] surprisingly reports that ASR performance is not a so good predictor of user's satisfaction.

Analytical Evaluation

The principal drawback of the method described above is, of course, the need of data collection. Indeed, subjective evaluation means that the system should be released to be evaluated. Although the usual process of SDS design obeys to the classical prototyping cycle composed of successive pure design and user evaluation cycles, there should be as little user evaluations as possible because it is time consuming and it is often very expensive. This is why some attempts to analyse strategies by mathematical means have been developed. In [Louloudis *et al*, 2001], the authors propose some way to diagnose the future performance of the system during the design process. Other mathematical models of dialogue have been proposed [Niimi & Nishimoto, 1999] and closed forms of dialogue metrics (like the number of dialogue turns) have been proposed. Nevertheless, too few implementations were made to prove their reliability. Moreover, lots of simplifying assumptions have to be made for analytical evaluation and it is thus difficult to extend those methods to complex dialogue configurations.

Computer-Based Simulation

Because of the data collection's inherent difficulties, some efforts have been done in the field of dialogue simulation. The purpose of using dialogue simulation for SDS evaluation is mainly to enlarge the set of available data and to predict the behaviour of the SDS in unseen situations. Simulation techniques will be more extensively discussed in the Second Part of this text.

II.5. Conclusion

A complete SDS relies on lot of different techniques manipulating high-level (see sections II.3.2, II.3.3, II.3.5 and II.4) and low-level data (see sections II.3.1 and II.3.4) going from acoustic signal to understood concepts. The design of such a system therefore implies lots of strategic choices on the implementation of each of the sub-systems. Moreover, given the sub-systems surrounding the DM, the internal functioning of this last sub-system should be adapted to the performance of the others. Thus the challenges of the design of an optimal DM include the adaptation to DSP and NLP modules' performance but also the choice of the knowledge representation (section II.3.5), the dialogue model (section II.4.1) and the dialogue management method (section II.4.2). Moreover, the strategy will differ dramatically according to the choice of the degree on initiative let to the user (section II.4.3) and the presence of error repair processes involving potential confirmation (section II.4.4). One main challenges of the strategy design is actually to find the optimal mixture of all the possibilities that will lead to maximize the user's satisfaction. That is to achieve the goal of the user without decreasing the ergonomics. For this purpose, evaluation of dialogue strategies in needed (section II.4.5).

It could be shown in all the previous sections that those choices are very task dependent. It is therefore a major challenge of SDS design to ensure the portability and reusability of previous work or at least to define generic methods for design of dialogue systems.

Chapter III:
Some Useful Artificial Intelligence Techniques

III.1. A Bit of History

Artificial Intelligence (AI) can be seen as the branch of computer science that aims at modelling aspects of human thought on computers or at trying to solve by computer any problem that a human can solve faster. Thus AI deals with knowledge representation and the faculty for a computer of finding creative solutions to problems and to generalize to unseen situation. It finds it sources in the development of two science fields, namely computer science and probability theory, which amazingly appeared at the same time. Indeed, the theory of probabilities was born from a series of problems posed by the Chevalier de Méré in 1654 to the philosopher and mathematician Blaise Pascal who was also the inventor of the first mechanical adding machine, the 'Pascaline'. The Chevalier de Méré was very interested by gambling and his problem was to decide whether or not to bet even money on the occurrence of at least one 'double 6' during 24 dice throws. This eternal problem of easy-earning money by chance led to an exchange of letters between Blaise Pascal and Pierre de Fermat in which the fundamentals of probability theory were formulated for the first time. Although lots of mathematicians and philosophers brought significant contributions to the development of probability theory, the name of the Reverent Thomas Bayes (1702-1761) should surely be cited. Indeed, his theory about conditional probabilities [Bayes, 1763] led to the famous 'Bayes rule', which is probably the most used rule in the development of statistical AI systems and serves as a base of what is called Bayesian Learning techniques.

It is impossible to talk about the development of computer science without citing George Boole who published his determining Boolean algebra in his book [Boole, 1854]. But what is also interesting about this theory is that it has been developed in order to model human's thought and is a first attempt to a mathematical approach to decision making.

The pioneers of what we really call AI and was born in the 1950's were principally Alan Turing (who defined the so-called Turing Machine in [Turing, 1937] and the Turing Test in [Turing, 1950]), John von Neumann (who introduced the concept of a stored program in 1945 and published a formal description of the game theory applied to economics in [von Neumann & Morgenstern, 1944]) and Claude Shannon (who is the father of

information theory [Shannon, 1948] and published articles about a chess player machine [Shannon, 1950 a], [Shannon, 1950 b]). Since a lot of theoretical contributions to AI have been made during the last 50 years, there are so many fields included into AI that it would not be possible to describe all of them here. Nevertheless, it can be noticed that since the 1950's lots of spectacular applications have been developed. Based on an idea of Alan Turing, Joseph Weizenbaum developed in 1966 is famous ELIZA program [Weizenbaum, 1966] already discussed in section II.1, which is one of the first significant popular contribution to AI in terms of an application. For the last decades, everybody could see on TV robot learning to walk, the Japanese company Sony releases new dog robots each year, video games have more and more learning capabilities. The most amazing application is probably the IBM supercomputer Deep Blue that beat the world chess champion Garry Kasparov in 1997 [Hsu, 2002].

III.2. Reinforcement Learning

Reinforcement Learning (RL) [Sutton & Barto, 1998] addresses two main problems of AI, namely unsupervised machine learning and decision-making. From this point on, the learner and decision-maker will be called the (RL) *agent* and all what is outside the agent is its *environment*. Actually, RL addresses the problem of learning how to act through trial-and-error interactions with a dynamic environment so as to maximize an overall reward. That is how to optimally map situations to actions by trying and observing environment's feedback. It can also be regarded as a way of programming AI agents by reward and punishment without needing to specify how the task has to be achieved. In the most challenging cases, actions may affect not only the immediate reward, but also the next situation and, through that, all subsequent rewards. Trial-and-error search and delayed reward are the two important features of RL. A good example of an RL task is chess playing: when a chess player makes a move, his decision is driven both by anticipating possible replies and by immediate, intuitive judgments of the desirability of particular positions. This can lead the player to perform a seemingly silly move (deliberately loose his queen, for instance) in order to obtain a greater reward later (winning the game). Notice that RL does not defines algorithms but a family of problems and each algorithm that permits to solve an RL problem will be called an RL algorithm.

RL differs from supervised learning in that there is no presentation of input/output pairs but instead, after choosing an action the agent can observe the immediate reward and the subsequent state but not which action would have been the best in its long term interest. Furthermore another significant difference is that online performance is important since the learner interacts with the environment during the learning process. Evaluation of the system is therefore often concurrent with learning.

Contributions to RL theory were made by researches leaded in two different fields: optimal control and trial-and-error learning, which started in the psychology of animal learning (the term of RL was actually borrowed from animal learning studies by Marvin Minsky in his PhD thesis [Minsky, 1954]). Optimal control study started in the 1950's with Richard Bellman's researches in which he developed the Dynamic Programming (DP) algorithm [Bellman, 1957 a] for solving recursively optimal control problems. Bellman's equation is still commonly used as a starting point for lots of RL algorithms. Nevertheless, it is only after Barto, Sutton and Anderson published their results about a first RL application to the complex pole-balancing problem [Barto *et al*, 1983] that RL started to be more widely studied. During the last 20 years, RL paradigm has been applied

to a wide range of problems with one of the most impressive application being that by Gerry Tesauro to the game of backgammon: the TD-Gammon [Tesauro, 1994]. It reached nearly the level of the world's strongest grandmasters.

Formally, solving an RL problem can be seen as a sequential process during which an agent and its environment interact at each of (not always equally distant) discrete time steps $t = t_0, t_1, t_2, t_3, \ldots$ (see Fig. 15). At each time step t (which will be called a *turn* from now on) the agent receives some observation about its environment's state $\mathbf{s_t} \in S = \{s_i\}$ (the state space), and accordingly selects an action $\mathbf{a_t} \in A = \{a_i\}$ (set of possible actions in state s_t). One turn later, partly as a consequence of its action a_t, the agent receives a numerical reward $\mathbf{r_{t+1}}$ and its internal state switches to $\mathbf{s_{t+1}}$.

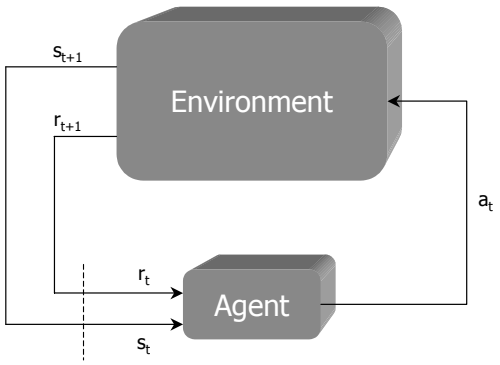


Fig. 15: RL as a sequential process

At each turn t, the agent realises a mapping from states to probabilities of performing each possible action. This mapping is the agent's *policy* or *strategy* $\boldsymbol{\pi_t}$, with $\pi_t(s, a)$ being the probability of performing action a in state s and $\pi_t(s) = a$ if the policy is deterministic. An RL algorithm defines how the agent updates its policy as a result of its experience. The agent's goal is then to maximize the expected total amount of rewards over the long run.

Thus, an optimal policy $\boldsymbol{\pi}^*$ would select at time t_i the action a that maximises a kind of additive function of rewards $\{r_{t+1}\}_{t>ti}$. This function is called the *return* $\mathbf{R_t}$. In the simplest case the return is just the sum of the rewards:

$$R_t = r_{t+1} + r_{t+2} + \ldots + r_T \qquad \text{(III.1)}$$

This expression of the return supposes that the interactions between the agent and the environment ends at time T (in the terminal state s_T). Such a task that ends at a time T is referred to as an *episodic task* and each run is called an *episode* while R_t is called the *undiscounted return*. Yet, if the interaction never ends, which is the case in *continual tasks*, the return could be infinite and thus, its maximisation could not serve as an objective function for learning an optimal strategy. This is why the concept of *discounted return* is introduced:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad \text{(III.2)}$$

with $0 \leq \gamma \leq 1$ called the *discount rate*. The discount rate determines the present value of future rewards. If $\gamma = 0$ the agent is only concerned with maximizing immediate rewards, as γ approaches 1, the objective takes future rewards into account more strongly. If the set of rewards is bounded, the discounted return is finite.

Most of RL algorithms are based on the estimation of two value functions for a given policy π: the *state-value function* $\mathbf{V^\pi}$ and the *action-value function* $\mathbf{Q^\pi}$. The value $V^\pi(s)$ of

a state s under the policy π, estimating how good it is to be in state s according to the policy π, is the expected return starting from state s and following policy π thereafter:

$$V^{\pi}(s) = E_{\pi}[R_t \mid s_t = s] \tag{III.3}$$

The value $Q^{\pi}(s,a)$ of performing action a in state s under the policy π, estimating how good it is to perform action a when in state s, is defined as the expected return starting from s, taking the action a, and thereafter following policy π:

$$Q^{\pi}(s,a) = E_{\pi}[R_t \mid s_t = s, a_t = a] \tag{III.4}$$

An optimal policy π^* is a policy resulting in a maximal value $V^*(s)$ for all states and an agent would follow this optimal policy by selecting in each state an action that have a maximal value $Q^*(s, a)$.

Since the learning process is often based on the computation of those values by means of trial-and-error cycles, one could easily understand that each state and each state-action pair should be visited an infinite number of times in order to estimate V^{π} and Q^{π} (this is called the *search process* or *exploration*). Yet, if the RL agent is actually interacting with a real environment, it should not deviate too much from a strategy that leads to an acceptable behaviour and thus should exploit what was already learned (this is called *exploitation*). A trade-off between exploration and exploitation should therefore be made. Several proposals have been made to face this problem and two of them are widely used: the *ε-greedy* action selection and the *softmax* action selection. When using the ε-greedy method, the learner selects actions according to the following policy:

$$a_t = \begin{cases} \underset{a}{arg\,max}\, Q^{\pi}(s_t, a) \text{ with probability } (1 - \varepsilon) \\ \text{any } a \in A \text{ with probability } \varepsilon / |A| \end{cases} \tag{III.5}$$

This means that the learner selects a non-greedy action with a probability ε and that each action of the set has an equal probability to occur in that case. The greater ε is, the more the learner explores. On another hand, the softmax method takes into account the value of each action to select the next one in order not to perform very bad actions too often. One way to implement the softmax action selection method is to use a Boltzmann or Gibbs distribution in order to compute the probability of occurrence on an action as a function of its value:

$$P(a_t = a \mid s_t = s) = \frac{e^{Q^{\pi}_{t-1}(s,a)/\tau}}{\sum_{b \in A} e^{Q^{\pi}_{t-1}(s,b)/\tau}} \tag{III.6}$$

The τ parameter is a positive real number called the *temperature*. When it has a high value all actions tend to be equally probable and when the temperature equals zero the learner always chooses the optimal action given the current policy, the one with a higher value. The temperature can be decreased along the time in order to make the learner less explorative after having learned an acceptable policy.

III.2.1. Solving RL Problems within the Framework of Markov Decision Processes

In the aim of solving problems by means of RL algorithms, a mathematical framework should be defined and particularly, the Markov property of the state signal should be assumed.

Markov Property

An RL problem satisfies the Markov property if the state and reward at time t+1 only depends on state and action at time t:

$$P(s_{t+1}, r_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, ..., s_1, a_1) = P(s_{t+1}, a_{t+1} \mid s_t, a_t) \tag{III.7}$$

This does not mean that the history of the interaction is irrelevant for decision-making at time t, this means that the state representation should be sufficiently well designed as to retain relevant information for decision-making purposes. A state signal that succeeds in retaining all relevant information is said to be Markov, or to have the Markov property.

MDP Definition

An RL problem that satisfies the Markov property is called a Markov Decision Process (MDP). An MDP is completely defined by a tuple $\{S, A, \mathcal{T}, \mathcal{R}\}$ where S is the state space, A is the action set, $\mathcal{T}$ is a transition probability distribution over the state space and $\mathcal{R}$ is the expected rewards distribution. The couple $\{\mathcal{T}, \mathcal{R}\}$ defines the one-step dynamics of the system:

$$\begin{aligned} \mathcal{T}^{a}_{ss'} &= P(s_{t+1} = s' \mid s_t = s, a_t = a) \\ \mathcal{R}^{a}_{ss'} &= E[r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'] \end{aligned} \tag{III.8}$$

An MDP can be described by a weighted node-transition network with two node types: state nodes and action nodes like depicted on Fig. 16. On this figure, state nodes are big empty circles while action nodes are small full circles. A weighted directed arc represents each transition with weights being the probability of the transition occurring. Rewards are also specified.

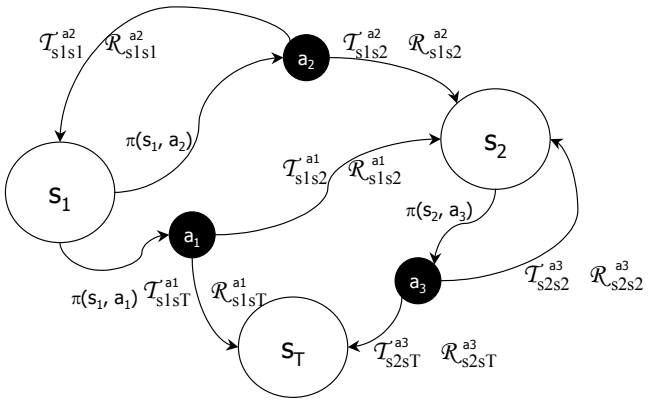


Fig. 16: A node-transition representation of MDP

According to this definition of MDPs, the state-value function can be rewritten:

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}[R_t \mid s_t = s] \\
&= E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right] \\
&= E_{\pi}\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right] \\
&= \prod_{a \in A} \pi(s,a) \sum_{s' \in S} \mathcal{T}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma E_{\pi}\left[\gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right]\right) \\
&= \prod_{a \in A} \pi(s,a) \sum_{s' \in S} \mathcal{T}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma V^{\pi}(s')\right)
\end{aligned}
\tag{III.9}
$$

In the same way, the action value function can also be rewritten:

$$
\begin{aligned}
Q^{\pi}(s,a) &= E_{\pi}[R_t \mid s_t = s, a_t = a] \\
&= E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right] \\
&= E_{\pi}\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a\right] \\
&= \sum_{s' \in S} \mathcal{T}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma E_{\pi}\left[\gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right]\right) \\
&= \sum_{s' \in S} \mathcal{T}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma V^{\pi}(s')\right)
\end{aligned}
\tag{III.10}
$$

The last equalities of expressions III.9 and III.10 are called the Bellman's equations for V^{π} and Q^{π}. Computing $V^{\pi}(s)$ for all states allows the evaluation of the policy π and an order among policies can be defined. Indeed, a policy π_1 is defined to be better than or equal to a policy π_2 if its expected return is greater than or equal to that of π_2 for all states. This means that, $\pi_1 \geq \pi_2$ if and only if $V^{\pi 1}(s) \geq V^{\pi 2}(s)$ for all states s. As the goal is to find the optimal policy $\pi^* \geq \pi \ \forall \pi$ that leads to maximal values $V^*(s)$ and $Q^*(s,a)$, one can write:

$$
\begin{aligned}
V^*(s) &= \max_{a \in A} Q^{\pi^*}(s,a) \\
&= \max_{a \in A} \sum_{s' \in S} \mathcal{T}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma V^{\pi}(s')\right)
\end{aligned}
\tag{III.11}
$$

$$
Q^*(s,a) = \sum_{s' \in S} \mathcal{T}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma \max_{a \in A} Q^*(s',a')\right) \tag{III.12}
$$

Those two last equations are the Bellman's optimality equations for V^* and Q^*.

Dynamic Programming

Assuming that all the parameters of the studied MDP are known and especially $\mathcal{T}$ and $\mathcal{R}$, equation III.9 defines a system of $|S|$ simultaneous linear equations in $|S|$ unknowns (the $V^{\pi}(s)$). That means that an exact evaluation of a policy π can be obtained by solving this system. Since the computation of the exact solution is possible but tedious, an iterative solution method is more suitable. It will also help for understanding the next sections.

During this iterative method, the Bellmann's equation for V^{π} (III.9) is used as an update rule:

$$V_{k+1}^{\pi}(s) = \prod_{a \in A} \pi(s,a) \sum_{s' \in S} \mathcal{T}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma V_{k}^{\pi}(s') \right) \tag{III.13}$$

This kind of operation is called a *full backup* since each iteration backs up the value of every states once to produce the new approximate value function. It is called a Dynamic Programming (DP) method because it is based on a perfect model of the environment, it eventually produces an exact solution and it is based on Bellman's equations, which expresses the problem of a maximisation process in terms of solving maximisation sub-problems (typically solved by a full backup process).

Once $V^{\pi}(s)$ has been computed, $Q^{\pi}(s,a)$ can be computed by using III.10. Remembering that $V^{\pi}(s)$ indicates how good it is to follow policy π from state s and $Q^{\pi}(s,a)$ indicates how good it is to take action a in state s and follow policy π thereafter, it is obvious that if $Q^{\pi}(s,a) \geq V^{\pi}(s)$, the policy should be updated in order to select a in state s ($\pi(s) = a$). Then, the second step of optimal policy learning (after evaluation) is policy *updating* or policy *improvement*:

$$\pi'(s) = \underset{a}{arg\,max}\, Q(s,a) \tag{III.14}$$

Once the policy has been updated, it is evaluated again and an iterative process composed of evaluation-improvement cycles starts until no significant improvement is noticed.

Monte Carlo Method

The term Monte Carlo is used to define a family of methods that provides approximate solutions to a variety of mathematical problems by performing statistical sampling experiments. It comes from the city in the Monaco Principality, because of a roulette, a simple random number generator and has been introduced in 1944 by physicists at Los Alamos who devised games of chance that they could study to help understand complex physical phenomena relating to the atom bomb. Here, it defines a family of methods in which the agent learns from interacting directly with the environment and averaging sampled returns. It is mainly defined for episodic task since the sampled returns are obtained at the end of each episode.

Using the Monte Carlo method allows evaluating the state value function $V^{\pi}(s)$ and the action value function $Q^{\pi}(s,a)$ by just measuring and averaging the return obtained from the first occurrence of s or (s,a). Nevertheless, what is of interest is the optimal control when interacting with the environment. Thus, after each episode the action value is updated and the strategy is modified:

$$\begin{aligned} Q^{\pi}(s,a) &= \text{average}(R(s,a)) \\ \pi'(s) &= \underset{a}{arg\,max}\, Q^{\pi}(s,a) \end{aligned} \tag{III.15}$$

It can be demonstrated that the Monte Carlo method converges toward the same values than the iterative DP method developed in the previous section and that after having visited all state-action pairs an infinite number of times, the learned strategy is optimal. The major drawback of the Monte Carlo method is that the policy is only updated once and the end of each episode and thus it converges slowly.

Temporal Difference

Temporal Distance (TD) learning [Sutton, 1988] combines ideas from Monte Carlo and DP methods. As Monte Carlo methods, TD methods learn from interactions without an exact model of the environment's dynamics. Like DP methods, TD methods update estimates based partly on other learned estimates, without waiting for the end of the episode (bootstrapping). Indeed, at time t+1 the agent builds a target for updating values based on the reward r_{t+1}.

The simplest TD method is the TD(0) method where the update is simply based on the immediate reward and the next value:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\left(r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)\right) \quad \text{(III.16)}$$

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\left(r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\right) \quad \text{(III.17)}$$

Whenever a state s in visited, its value (and the action value) is immediately updated to be closer to $r_{t+1} + \gamma V_t(s_{t+1})$ (or to $r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1})$ for the action value), which is the target for evaluation by TD(0) method. In III.16 and III.17, α is a positive step-size parameter called the *learning rate*. It can also be shown that the TD(0) method converges to the optimal values after each state-action pair has been visited an infinite number of times. The TD(0) method that implements the update rule III.17 is called the SARSA algorithm since it is based on the tuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$.

In reality, TD(0) is an instance of a more general class of algorithms called TD(λ). Actually, TD(0) only uses information from one step ahead to update values. The TD(λ) methods have 'more memory' and the update rule is:

$$V_{t+1}(s) = V_t(s) + \alpha\left(r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)\right) \cdot e_t(s) \quad \forall s \in S \quad \text{(III.18)}$$

In this expression, $e_t(s)$ is the *eligibility trace* of state s at time t. Thus, all the state values are updated at each turn (the same thing happens for action values). The eligibility trace is defined as follow:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t \\ \gamma\lambda e_{t-1}(s) & \text{otherwise} \end{cases} \quad \text{(III.19)}$$

with $0 \le \lambda \le 1$. One can see that $e_t(s)$ decays exponentially with time when state s is not visited. This means that the importance of the reward r_{t+1} obtained in time t+1 when stepping from state s_t to state s_{t+1} on the update performed on the value of state s decreases as the last time that s as been visited is far away from t. The same type of update rule can be defined for the action values in order to obtain the SARSA(λ) algorithm:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left(r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\right) \cdot e_t(s,a)$$

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases} \quad \text{(III.20)}$$

The policy is then updated by either choosing the action with a maximum value or by via an ε-greedy strategy or a softmax strategy using the updated action values. Of course, TD(λ) methods converge more quickly than Monte Carlo or TD(0) methods since the immediate reward serves to update all the state and action values for already visited state-action pairs. Notice that when $\lambda = 0$, the method becomes the TD(0) method and when $\lambda = 1$, the method becomes a kind of Monte Carlo method with discounted return.

Q-Learning

One of the most important breakthroughs in RL was the development of an off-policy TD control algorithm that learns the optimal policy while following another for control, known as the Q-learning method [Watkins, 1989]. The simplest update rule for Q-learning (1-step Q-learning) is defined by:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q_t(s_{t+1},a) - Q_t(s_t,a_t)\right) \quad \text{(III.21)}$$

It can be demonstrated that, whatever the action selected in state s_t (thus, whatever the followed policy), the evaluated policy is the optimal one. That means that whatever the policy being followed, the update rule will lead to evaluate $Q^*(s,a)$ for all state-action pairs if each pair is visited a sufficient number of times. The eligibility trace can be introduced in the framework of Q-learning, leading to the Q(λ) methods. Two different methods are known: Watkins' Q(λ) and Peng's Q(λ) [Peng & Williams, 1994]. The Peng's method is more difficult to implement and does not guarantee to converge (although experimental results proved quick convergence on several tasks), thus we only describes the Watkins method. According to Watkins, the update rule and the eligibility trace are:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left(r_{t+1} + \gamma \max_a Q_t(s_{t+1},a) - Q_t(s_t,a_t)\right)\cdot e_t(s,a) \quad \text{(III.22)}$$

$$e_t(s,a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t \\ 0 & \text{if } s = s_t \text{ and } a \neq a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases} \quad \text{(III.23)}$$

This method is quite easy to implement and proved to converge while enabling to follow an acceptable policy during learning.

Factored MDP

A particular type of MDPs is *factored MDPs* [Boutilier *et al*, 1999] in which each state and action is represented as a set of variables, namely *state variables* and *action variables*. Formally, a factored MDP is fully determined by the tuple $\left\{\left\{S^\alpha\right\}_{\alpha=1}^{M}, \left\{A^\beta\right\}_{\beta=1}^{N}, \mathcal{T}, \mathcal{R}\right\}$ in which:

- $\mathcal{T}$ and $\mathcal{R}$ are defined as previously,
- S^α is the set of all possible values for state variable α
- A^β is the set of all possible values for action variable β

A state is then a M-tuple and an action is a N-tuple and it is possible to take advantage of state and action variable independency in order to solve MDP problems with other techniques or even to learn the topology of the MDP. Indeed, by using a factored representation for states and actions, it is possible that certain action variables affect only certain state variables and the transition probabilities are more easily estimated.

Dialogue as an MDP

The MDP framework is suitable for representing a HMD since it can reasonably be considered as a sequential process. Indeed, a dialogue is defined as a turn taking interaction between two interlocutors. Moreover, each participant performs actions by

uttering sentences (Speech Acts) and modifies his state by updating his knowledge. Actions are taken according to a more or less defined strategy as far as we consider goal-directed dialogues. The formalisation of an SDS in the framework of MDPs as been firstly described in [Levin *et al*, 1997] and [Levin *et al*, 1998]. As explained later, a HMD can also be seen as a factored MDP under certain assumptions. More details will be provided in the Third Part of this text.

Partially Observable Markov Decision Processes

Although it can be argued that a HMD can be modelled as an MDP, it is important to remember that one of the dialogue's participants is a machine that infers human user's intentions from noisy spoken utterances recognized by an imperfect ASR system and using a possibly error prone NLU. Thus, the inferred dialogue state is not always accurate. This can be seen as an uncertainty about the state given the observation and the process might not be Markov any more in the observations. Another paradigm is then defined to characterise this kind of problems: Partially Observable Markov Decision Process (POMDP) [Sondik, 1971]. This approach assumes that the process is still Markov in some state variables that cannot be observed and the observation at time t are then assumed to be conditionally dependent on a hidden state at time t. Given this hidden state, observation at time t is also assumed to be independent of all other hidden or observable variables. The unobserved Markov process is called the *core process*. A POMDP is then defined by a tuple $\{A, S, O, \mathcal{T}, \mathcal{R}, O\}$ with:

- A, S, $\mathcal{T}$, $\mathcal{R}$, being the same as previously
- $O = \{o_i\}$ being the set of observations
- O being the observation function that gives for each action and resulting state a probability distribution over possible observations. It actually gives the probability of making observation o given that the action performed at time t was a and that it leaded to state s': $O^a_{s'o}$

A POMDP can also be factored using the same definitions as previously and by adding a set of observation variable values $\left\{O^\delta\right\}_{\delta=1}^{K}$ to the other sets.

Several methods for solving POMDPs have been proposed including exact solutions like DP methods and mainly approximate solutions because solving POMDPs is very complex. In this framework, a HMD can be defined as a POMDP by stating that the error prone recognised sentences are observations and hidden states are (partly) built on the user's real intentions. This approach has been firstly studied in [Roy *et al*, 2000] and the authors could demonstrate that the POMDP approach outperforms the pure MDP methods in the case of a dialogue management for a robot. Yet, the computational cost was very high, they obtained a sub-optimal solution and they had to reduce drastically the size of the state space. Indeed, POMDP solution is intractable for sate spaces larger than 15 states, which is a strong limitation. Another experiment is described in [Zhang *et al*, 2001]. Although this research shows once again that the POMDP solution, even if sub-optimal, outperforms the MDP one, it faces the same problems as the previous research. Moreover, it points out that the design of the POMDP is an expert job and is very task-dependent especially for the estimation of the observation function. All those problems make impossible the use of POMDPs in the framework of a methodology for task-independent design of SDSs.

III.3. Bayesian Networks

A Bayesian Network (BN) also called Belief Network is a directed acyclic graph encoding relationships among variables in a probabilistic framework [Pearl 1988]. In such a graph, nodes represent stochastic variables and arcs represent dependencies between variables. It creates an easy and readily way to encode uncertain expert knowledge of a given domain. A network so constructed also reflects implicit independencies among the variables by the lack of arcs. The network must be quantified by specifying a probability for each variable conditioned on all possible values of its immediate parents in the graph. In addition, the network must include a marginal distribution encoding unconditional probabilities for each variable that has no parent. This quantification is realised by associating a Conditional Probability Distribution (CPD) at each node. If the variables are discrete, this can be represented as a table (Conditional Probability Table: CPT), which lists the probability that the child node takes on each of its different parent set values. Together with the independence assumptions defined by the graph, this quantification defines a unique joint distribution over the variables in the network. Then, exploiting the independencies represented within the graphical structure, the probabilistic inference aiming at computing the probability of any event over this space can be realised.

The easier way to understand BN is to consider an example, so let's have a look to the following situation taken form [Murphy, 2001]. A person walks by a garden and can see that the grass is wet. Since a water sprinkler is installed in the middle of the grass, there can be two reasons for the grass being wet: either the water sprinkler was on or it has been raining. On another hand, the water sprinkler is more often switched on when the sky is sunny (it might even be automated) and it is more likely to rain when the sky is cloudy. According to this prior knowledge of the domain, what is the most likely reason for the grass being wet?

This example can be captured in the following BN in which prior knowledge about conditional probabilities have been encoded:

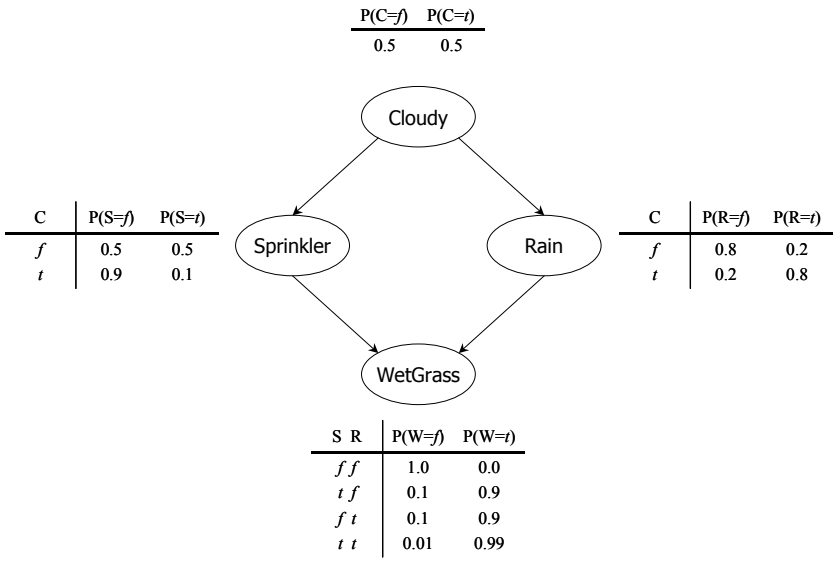


Fig. 17: The water sprinkler BN

In this example, four Boolean variables are defined (C for cloudy sky, S for sprinkler on, R for rainy weather, W for wet grass). Before going further, one should notice that all conditional probabilities are given while only half of them are really needed since, according to the sum rule of probabilities, each row must sum to 1. Thus, nine parameters are needed to completely define the BN. Since all variables are Boolean in this example, the complete distribution of variables over the domain is defined by the 2^4-1 = 15 joint distributions P(C, S, R, W). The saving doesn't seem great but considering the network topology of Fig. 18, where all variables are still Boolean, one can see that 2^{12}-1 joint distributions should be needed to define the domain while only 31 parameters are needed in the BN framework. This is why BNs are often considered as a compact representation of the probabilistic properties of a domain. This reduction is done thanks to implicit variable independence assumptions made when building the network.

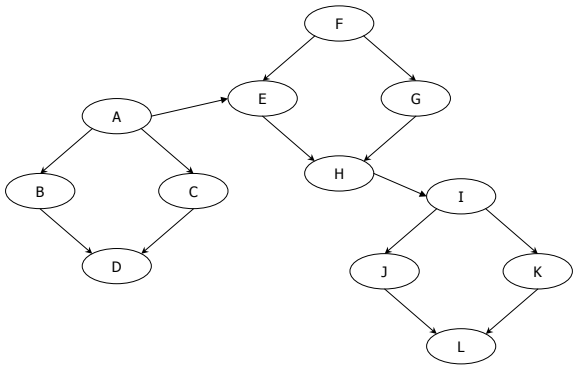


Fig. 18: A 12-variables BN

Actually, the BN framework makes use of both probability theory and graph theory to define conditional independency between variables. Graph theory can help in deriving conditional independency between variables easily and without make use of the Bayes' rule. Indeed, according to graph theory, two variables a and b are conditionally independent given an evidence e, that is P(a | b, e) = P(a | e), if a and b are *d-separated* given e (or if e *d-separates* a and b, or they are not *d-connected* given e or there isn't a *d-connection* path between a and b given e). More generally, a set of variables A is conditionally independent of a set of variables B given a set of evidences E if E d-separates A and B. A and B are d-separated given E if along every undirected path between a node a$\in A$ and a node b$\in B$ there exist a node d such that:

- d has converging arrows and neither d or any of its descendents belongs to E
- d does not have converging arrow and d$\in E$

In the sprinkler water BN for example, W is conditionally independent from C given the evidence set E= {R, S} since R and S are along the path going from C to W and neither R nor S have converging arrows. That means that knowing that it is raining for instance, the state of the grass is independent of the state of the sky. In the same way, R is conditionally independent from S given the evidence E=C since the upper path going from R to S goes through C and the lower path contains W that has convergent arrows but W doesn't belong to E.

III.3.1. Probabilistic Inference

The most common task one wish to solve using BNs is probabilistic inference. That is finding the likelihood of a variable state given some evidence. For example, considering the network of Fig. 17, and suppose that the grass is actually wet. There are two possible causes: either it has been raining or the sprinkler was just on. Which is more likely? Let's use the Bayes' rule to compute the posterior probability of each possibility:

$$\mathrm{P}(\mathrm{S}=t \mid \mathrm{W}=t)=\frac{\mathrm{P}(\mathrm{S}=t,\mathrm{W}=t)}{\mathrm{P}(\mathrm{W}=t)}=\frac{\sum_{\mathrm{c,r}}\mathrm{P}(\mathrm{C}=\mathrm{c},\mathrm{R}=\mathrm{r},\mathrm{S}=t,\mathrm{W}=t)}{\sum_{\mathrm{c,r,s}}\mathrm{P}(\mathrm{C}=\mathrm{c},\mathrm{R}=\mathrm{r},\mathrm{S}=\mathrm{s},\mathrm{W}=t)} \tag{III.24a}$$

$$\mathrm{P}(\mathrm{R}=t \mid \mathrm{W}=t)=\frac{\mathrm{P}(\mathrm{R}=t,\mathrm{W}=t)}{\mathrm{P}(\mathrm{W}=t)}=\frac{\sum_{\mathrm{c,s}}\mathrm{P}(\mathrm{C}=\mathrm{c},\mathrm{R}=t,\mathrm{S}=\mathrm{s},\mathrm{W}=t)}{\sum_{\mathrm{c,r,s}}\mathrm{P}(\mathrm{C}=\mathrm{c},\mathrm{R}=\mathrm{r},\mathrm{S}=\mathrm{s},\mathrm{W}=t)} \tag{III.24b}$$

Inference is based on the computation of the joint probability, which can be easily computed thanks to independence assumptions made when building the network:

$$\begin{aligned}\mathrm{P}(\mathrm{W},\mathrm{S},\mathrm{R},\mathrm{C}) &= \mathrm{P}(\mathrm{W} \mid \mathrm{S},\mathrm{R},\mathrm{C})\cdot\mathrm{P}(\mathrm{R} \mid \mathrm{S},\mathrm{C})\cdot\mathrm{P}(\mathrm{S} \mid \mathrm{C})\cdot\mathrm{P}(\mathrm{C}) \\ &= \mathrm{P}(\mathrm{W} \mid \mathrm{S},\mathrm{R})\cdot\mathrm{P}(\mathrm{R} \mid \mathrm{C})\cdot\mathrm{P}(\mathrm{S} \mid \mathrm{C})\cdot\mathrm{P}(\mathrm{C})\end{aligned} \tag{III.25}$$

Because of this, a BN can also be seen as a graphical way to represent a particular factorisation of a joint distribution. According to this factorisation one can write, for instance:

$$\begin{aligned}\mathrm{P}(\mathrm{W}=\mathrm{w}) &= \sum_{\mathrm{c}}\sum_{\mathrm{s}}\sum_{\mathrm{r}}\mathrm{P}(\mathrm{C}=\mathrm{c},\mathrm{R}=\mathrm{r},\mathrm{S}=\mathrm{s},\mathrm{W}=\mathrm{w}) \\ &= \sum_{\mathrm{c}}\sum_{\mathrm{s}}\sum_{\mathrm{r}}\mathrm{P}(\mathrm{C}=\mathrm{c})\cdot\mathrm{P}(\mathrm{R}=\mathrm{r} \mid \mathrm{C}=\mathrm{c})\cdot\mathrm{P}(\mathrm{S}=\mathrm{s} \mid \mathrm{C}=\mathrm{c})\cdot\mathrm{P}(\mathrm{W}=\mathrm{w} \mid \mathrm{S}=\mathrm{s},\mathrm{R}=\mathrm{r}) \\ &= \sum_{\mathrm{c}}\mathrm{P}(\mathrm{C}=\mathrm{c})\sum_{\mathrm{s}}\mathrm{P}(\mathrm{S}=\mathrm{s} \mid \mathrm{C}=\mathrm{c})\sum_{\mathrm{r}}\mathrm{P}(\mathrm{R}=\mathrm{r} \mid \mathrm{C}=\mathrm{c})\cdot\mathrm{P}(\mathrm{W}=\mathrm{w} \mid \mathrm{S}=\mathrm{s},\mathrm{R}=\mathrm{r})\end{aligned} \tag{III.26}$$

All this provides the desired results:

$$\begin{aligned}\mathrm{P}(\mathrm{S}=t \mid \mathrm{W}=t) &= 0.430 \\ \mathrm{P}(\mathrm{R}=t \mid \mathrm{W}=t) &= 0.708 \\ \mathrm{P}(\mathrm{W}=t) &= 0.6471\end{aligned} \tag{III.27}$$

Thus, there is a probability of 0.6471 of the grass being wet, and if it is wet it is more likely because it has been raining. This process is called *bottom-up* inference or *diagnostic* process since that according to some observed evidences, the more likely cause has been inferred.

Notice that even if they are not defined as directly dependent in the network, S and R compete to explain W. Thus, one can for example compute the probability of the sprinkler being on while observing that it is raining and the grass is wet:

$$\mathrm{P}(\mathrm{S}=t \mid \mathrm{R}=t,\mathrm{W}=t)=0.1945 \tag{III.28}$$

The probability of the water sprinkler to be on decreases. It is known as Berkson's paradox. It also explains why BNs are also called belief networks since observing that the grass is wet but it has been raining decreases the belief that the sprinkler was on.

Finally, we can also compute causal effect of the state of the sky on the grass. That is, for instance, the probability of the grass being wet when the sky is cloudy:

$$P(W = t \mid C = t) = \frac{P(C = t, W = t)}{P(C = t)} = \frac{\sum_{r,s} P(C = t, R = r, S = s, W = t)}{P(C = t)} = \frac{0.3726}{0.5} = 0.7452 \quad \text{(III.29)}$$

This is called *causal* or *top-down* reasoning. Because of this particularity, BNs are also considered as generative models as they specify how causes generate effects. A generative model is generally defined as a statistical model which, when trained on a particular data set, can generate new data. They duplicate the probability distribution from which the original data were drawn so that they are able to generate new data points that are statistically similar to those on which the model was trained. Training of BNs will be briefly explained later.

In the simple case of the water sprinkler BN, exact inference was quite simple but it is not always the case. Indeed, the computation of exact inference for all nodes is NP-hard [Cooper, 1990]. Yet, exact inference algorithms exist for a special case of networks, which are the *singly connected* networks. In this special case, there is no loop in the network and an efficient algorithm, called belief propagation can do the job [Pearl 1988]. For *multiply connected* networks (like the water sprinkler BN), the solution must generally be approximated.

Some methods propose to transform multiply connected networks into singly connected networks. Other famous methods are based on the sampling of the network, that is they randomly posit values for some of the variables and then use them to pick up values for the other nodes. They then keep statistics on the values that the nodes take and these statistics give the answer. As said before, sampling methods are generally called Monte Carlo methods and they have the advantage of not being dependent of the network's topology. Monte Carlo methods include the particular case of Gibbs Sampling [Mackay, 1999]. Finally, other approximate methods exist such as the variational methods [Jourdan *et al*, 1999].

III.3.2. Learning Networks

Until now, it has been assumed that the whole network was defined, not only its topology but also the conditional probabilities through the parameters of CPD or CPT. In their early uses, BNs were used to encode experts knowledge about a particular domain. Experts drew the network and provided associated conditional probabilities according to their beliefs about the particular task and their confidence in these beliefs. In the mid 1990's AI researchers considered the possibility to learn parameters from data and structure as well. Thus, BNs can be considered as a combination of prior knowledge and statistical data. It is another reason to call this framework 'Bayesian' networks since probabilities are really considered as beliefs that can be updated by real data (and not as physical observation frequencies).

From now on, it will be considered that the studied domain can be encoded in a network B. Several situations can be investigated according to the observability of the domain variables and the knowledge about the structure. Although there exist several methods for

learning BNs, the focus will be on the Bayesian approach [Heckerman, 1995] to give an overview of possibilities offered by the BN framework.

Known Structure, Full Observability

Here it is supposed that the hypothetic structure B^h containing N nodes drawn by experts is accurate and that in addition, we have a data set (or database or training set) $D = \{d^1, .., d^M\}$ containing no missing data, that is each case d^m in the data set consists of an observation of all the variables of the domain. Finally, let's assume that all variables can only take discrete values. The goal of learning in this case is to find the values of the parameters of each CPD that maximizes the likelihood of the training data: $P(D|B^h)$. Actually, this is not complete since the experts' prior knowledge K of the domain should be mentioned. The likelihood of the training set is actually $P(D|B^h, K)$ and $P(\Theta_B|B^h,K)$ encodes the uncertainty of the experts about the prior parameters. It is the prior distribution of the parameters. First, let's consider the following definition:

- $V = \{v_1, \ldots, v_N\}$ is the set of domain variables
- each variable v_i can only have r_i values
- p_i is the set of parent nodes of v_i
- p_i can have $q_i = \prod_{x_l \in p_i} r_l$ distinct states denoted p_{ij}
- θ_{ijk} is the physical probability of v_i=k when p_i=j
- $\Theta_{ij} = \bigcup_{k=1}^{r_i} \{\theta_{ijk}\}$ is the set of parameters of variable i
- $\Theta_B = \bigcup_{i=1}^{N} \bigcup_{j=1}^{q_i} \Theta_{ij}$ is the set of parameters of the network.

In the case of the water sprinkler:

$V = \{C, S, R, W\}$ $\quad\quad$ $\theta_{RCt} = 0.8$

$r_i = 2\ \forall i$ $\quad\quad$ $\Theta_{RC} = \{P(R=t|C=t), P(R=t|C=f),$

$p_W = \{R, S\}$ $\quad\quad$ $P(R=f|C= t), P(R=f|C=f)\}$

Thus, the interest is to find the posterior distribution of the parameters Θ_B of the network given a prior distribution $P(\Theta_B|\ B^h, K)$ (which is a function of the knowledge) and a database D. Assuming independence of the learned parameters we have:

$$P(\Theta_{ij} \mid D, B^h, K) = \frac{P(D \mid \Theta_{ij}, B^h, K) \cdot P(\Theta_{ij} \mid B^h, K)}{P(D \mid B^h, K)} = \alpha \cdot \prod_{k=1}^{r_i} \theta_{ijk}^{M_k} \cdot P(\Theta_{ij} \mid B^h, K) \quad \text{(III.30)}$$

where N_k is the number of times that $v_i = k$ in the database and α is a normalisation constant. In general, it is assumed that the prior knowledge about the parameters has a Dirichlet distribution:

$$P(\Theta_{ij} \mid B^h, K) = \beta \cdot \prod_{k=1}^{r_i} \theta_{ijk}^{M'_{ijk}-1} \quad \text{(III.31)}$$

The posterior distribution of the parameters is then also Dirichlet:

$$P\left(\Theta_{ij} \mid D, B^h, K\right) = \gamma \cdot \prod_{k=1}^{r_i} \theta_{ijk}^{M_k + M'_{ijk} - 1} \tag{III.32}$$

Thus, everything looks like the prior parameters were drawn from a database containing M' cases in which there were M'_{ijk}-1 cases where $v_i = k$ and $p_i = j$ and that the new database in which there are M'+M cases and in which there is $M_k + M'_{ijk}$-1 such cases is considered. Assuming Dirichlet distribution of prior knowledge is therefore equivalent to add artificial pseudo counts to the empirical counts. M' and M'_{ijk} should be assessed by experts, it is a way to express their confidence in their prior knowledge.

This way to update parameter distribution is called the Maximum a Posteriori (MAP) method. Assuming no prior knowledge, the additional pseudo-counts are dropped and the parameters' estimates are equivalent to the Maximum Likelihood (ML) estimates, which are simply normalised counts of each setting of variables given each setting of its parents in the database.

Known Structure, Partial Observability

In this section, the case of a database *D* containing cases in which some of the domain variables are not observable (hidden variables) is considered. Like in other learning processes, the Expectation-Maximization (EM) can be used to learn maximum likelihood parameters of a generative model where some of the random variables are observed, and some are hidden [Dempster *et al*, 1977]. The hidden variables might represent quantities that are thought to be the underlying causes of the observations. For example, a model designed to explain data consisting of shoe sizes and reading ability might use age as a hidden variable.

As the name implies, there are two stages in the EM algorithm: in the E step, the expected values of all the nodes is computed using an inference algorithm, and then these expected values are treated as though they were observed to learn the new parameters by maximizing likelihood (M step). With *O* denoting the set of observable variables, *H* the set of hidden variables and Θ_B the current parameters of the network:

- E step: compute the distribution $P(H \mid O, \Theta_B)$ over the hidden variables, given the observation and the current value of the parameters. This is actually an inference problem.
- M step: compute the new parameters that maximizes the expected log-likelihood under the distribution found in the E-step:

$$\begin{aligned} \hat{\Theta}_B &= \underset{\Theta_B}{arg\,max} \left\{ E\left[log\, P\left(O, H \mid \Theta_B\right) \right]_{P(O|H)} \right\} \\ &= \underset{\Theta_B}{arg\,max} \sum_H P\left(O \mid H\right) \cdot log\, P\left(O, H \mid \Theta_B\right) \end{aligned} \tag{III.33}$$

The M-step might require solving a difficult non-linear optimisation problem and several exact and approximate solutions can be investigated. The iteration of those two steps has proved to converge to a local maximum under certain conditions [Dempster *et al*, 1977].

Unknown Structure

Sometimes, it can also be considered that prior knowledge is not only inaccurate about the parameters but also about the actual structure of the network B that better encodes the relationship between variables of the domain. Thus, before trying to learn a structure, a

key point is to establish a measure of how well a given hypothesis B^h about the structure fits the prior knowledge K and the data set *D*. A criterion often used as an objective function for learning structure is the likelihood of the model given K and *D*:

$$P(B^h \mid D, K) = \frac{P(D \mid B^h, K) \cdot P(B^h \mid K)}{P(D \mid K)} = \alpha \cdot P(D \mid B^h, K) \cdot P(B^h \mid K) \qquad \text{(III.34)}$$

Or more often the log-likelihood of the hypothesis on the data:

$$L = log\, P(D \mid B^h, K) + log\, P(B^h \mid K) + log\, \alpha \qquad \text{(III.35)}$$

The best network is the one that maximises *L*. The presence of the prior knowledge in the objective function is quite important since it avoids the maximisation process to assess one node to each case of the database on the learned structure. There are lots of methods to compute either exactly or approximately the terms of expression III.35 and particularly, the log marginal likelihood $P(D \mid B^h, K)$:

$$P(D \mid B^h, K) = \int P(D \mid \Theta_B, B^h, K) \cdot P(\Theta_B \mid B^h, K) d\Theta_B \qquad \text{(III.36)}$$

The Bayesian Information Criterion (BIC), for instance, can be used to approximate this term and is a trade off between a term measuring how well the parameterised model predicts the data and a term that punishes the complexity of the model [Schwartz, 1978]. Nevertheless, the priors on the parameters $P(\Theta_B \mid B^h, K)$ and on the structure $P(B^h \mid K)$ have to be known for the learning process. Several methods can also be considered and a good discussion about this can be found in [Heckerman, 1995].

Finally, having an objective function, we need search methods for identifying structures with high score by some criterion. This problem is, once again, a NP-hard problem since the number of directed acyclic graphs on N variables is super-exponential in N. The usual approach is therefore to use local search algorithms (for instance, greedy hill climbing, possibly with multiple restarts) or perhaps branch and bound techniques, to search through the space of graphs. Actually, the objective function can be decomposed as a product of local terms:

$$P(D \mid B^h, K) = \prod_{i=1}^{N} \int P(v_i = d_i^m \mid \Pi_i, \theta_i, K) \cdot P(\theta_i \mid K) d\theta_i \qquad \text{(III.37)}$$

This makes local search more efficient, since to compute the relative score of two models that differ by only a few arcs, it is only necessary to compute the terms that they do not have in common.

III.3.3. Some Special Cases

Introduction

Although BNs have only been developed as a general framework for a little more than two decades, some of their particular cases are widely used in practice for decision theory, pattern matching or data fusion. Here are some examples of BNs particular cases.

Influence Diagrams

Influence diagrams were originally introduced in [Howard & Matheson, 1984] as a compact representation of symmetric decision trees, but may also be considered as an

extension of BNs. Actually, we can see influence diagrams as BNs with three different kinds of nodes: belief nodes (as usual), utility nodes and decision nodes.

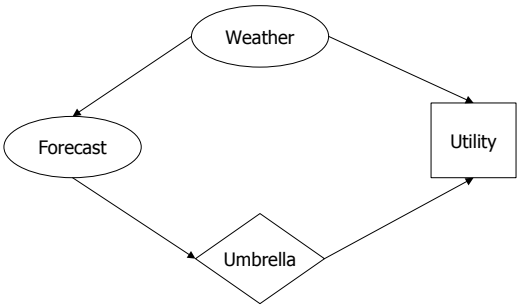


Fig. 19: Influence Diagram

The influence diagram shown on Fig. 19 depicts the problem of deciding whether or not taking an umbrella to go out according to the weather forecast. The actual weather influences the forecast and the utility of having taken the umbrella. On this figure, the belief nodes are ovals, action nodes are diamonds and utility nodes are squares. The problem of solving the influence diagram can be seen in two ways. First, the simple inference in which the problem is to find the utility of an action according to some observations (what's the utility of taking the umbrella while the forecast predicted cloudy weather and it is actually not raining). Second, the decision-making problem in which one wants to find the action maximising the expected utility according to some observations. Some variables might not be observable (hidden variables). The most widely used application of influence diagrams is probably the Lumiere project, which is embedded in the MS Office assistant [Horvitz *et al*, 1998].

Dynamic Bayesian Networks

Until now, it was considered that the BNs represented relationship between variables of a domain and could be learned from a database containing observation of some configuration of the variable set. The cases contained in the database were also considered independent. In temporal processes, each observation is a sequence of configurations of domain variables and each configuration (state) at time t influences the configuration (state) at time t+1. There is a temporal dependency among variables. Dynamic Bayesian Networks (DBNs) is a framework for representing temporal processes. They are directed graphical models of stochastic processes and represent the hidden (and observed) states in terms of state variables, which can have complex interdependencies. The graphical structure provides an easy way to specify these conditional independencies, and hence to provide a compact parameterisation of the model. It is generally assumed that the parameters do not change and the model is time-invariant.

On Fig. 20, a fully connected DBN is depicted. In general, temporal dependencies are described by drawing a 2-time slices graph (2TBN), since it is assumed that parameters are time-invariant. On this figure, empty circles represent observed variables while filled circles stand for hidden variables.

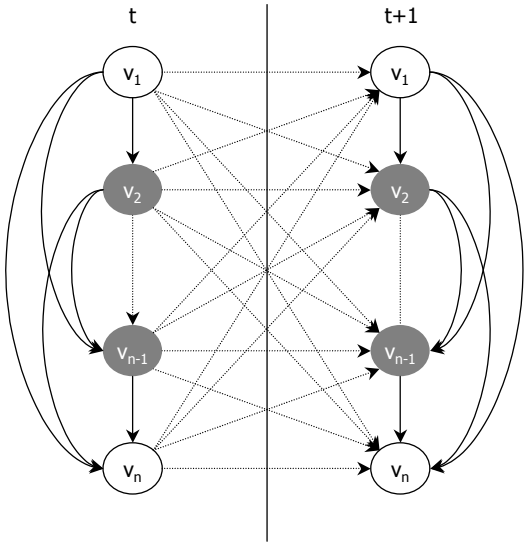


Fig. 20: 2 time slices of a fully connected DBN.

Two particular kinds of DBNs are widely used in AI researches and applications: the Hidden Markov Models (HMMs) extensively used in pattern matching (and specially in speech recognition: see section II.3.1) and the Linear Dynamic Systems (LDSs) of which Kalmann filters are special cases (widely used in data fusion). In the HMM framework for instance, a sequence of observed variables (X) is supposed to be conditioned by some hidden variables (Y) and the sequence of hidden variables is assumed to be Markov (see Fig. 21).

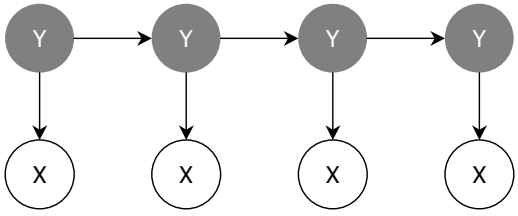


Fig. 21: HMM topology

Second Part

~

SIMULATING DIALOGUES

Although it is admitted that the general design process of human-machine interfaces is a cyclic process composed of prototype releases, user satisfaction surveys, bug reports and refinements, it is also well known that human intervention for testing is time-consuming and above all very expensive. Indeed, we live in an era of consumer surveys where appeared professional testers because people commonly get tired of answering questions. Avoiding human intervention as much as possible is then a desired objective that simulation tries to achieve. Although it is not (yet) realistic to simulate human behaviour in all situations, it is argued that human beings adapt their behaviour when communicating with machines and that this modified behaviour can be captured in probabilistic models. While Artificial Intelligence tried for a long time to simulate human behaviour in a wide range of applications, it finally looks like it was easier to get human to adapt to machines and find it natural… Yet, to encode this modified behaviour, most of probabilistic models need real examples, a knowledge database collected by prototype releases or by other means in order to generate new data points, which is precisely what was to avoid.

On another hand, a dialogue system relies on signal processing and natural language processing systems. A human-machine dialogue simulation should then take into account these sub-systems because they are prone to errors by nature. The optimal behaviour of the dialogue system as well as the user's behaviour will be dependent of performance of these elements. Models of task-dependent DSP and NLU performance should also be part of a realistic simulation environment.

Chapter IV:
Generalities about Dialogue Simulation

IV.1. Dialogue Simulation: How and What for?

First of all, let's define dialogue simulation as a method to artificially generate interactions between two agents engaged in a task-oriented dialogue (remember that a dialogue is an interaction between two agents based on sequential turn taking). The simulation purpose is of course to avoid releasing an imperfect system to users in order to collect data because real human tests are very expensive. Two ways of achieving this mission can be investigated: 'Wizard of Oz' simulation and computer-based simulation.

IV.1.1. Simulation Methods

The 'Wizard of Oz' (WOZ) method (or Oz Paradigm), developed in the early 80's by Jeff Kelley at the John Hopkins University [Kelley, 1984], is a testing or iterative design methodology wherein a human experimenter (the "Wizard") simulates the behaviour of a theoretical intelligent computer application and interacts with potential users. It takes its name from the obvious parallels with the 1900 book by L. Frank Baum. In the SDS framework, the human experimenter embodies the DM strategy and interacts with several human users for testing it. The WOZ technique is widely used for testing and designing NLP systems [Dahlbäck *et al*, 1993] and particularly for SDS evaluation and design [Klemmer *et al*, 2000]. The advantages of WOZ techniques are that no real implementation of the prototype is needed and real potential users can rate the system. Nevertheless, the purpose of this thesis is to avoid as much as possible human intervention during the design process.

One alternative to WOZ techniques is computer-based simulation. In this case, in opposition to WOZ method, the whole simulation system is composed of computer software. In this simulation framework, the current DM implementation (the DM prototype) interacts with another software standing for the DM's environment. The environment part of the software will be described later (see section IV.2) but it is obvious that it should include user simulation. Computer based simulation has the big advantage to avoid human interactions with a non-optimal system (the SDS prototype),

which can be very time consuming and very expensive. Another advantage of computer-based simulation over WOZ is that the actual current implementation of the DM is tested and not a human interpretation of the strategy.

IV.1.2. Computer-Based Simulation Application

There are several reasons for which dialogue simulation became the subject of lots of researches since thirty years. Initially, systems developed in the purpose of simulating dialogues aimed at validating dialogue models like those discussed in section II.4.1. They were essentially systems involving two artificial agents built according to a formal description of the model and communicating with each other. Their simulated exchanges were logged and, obviously, the more the logs looked like a real human-human dialogue, the better the model was (or at least, its formal description) [Power, 1979].

During the SDS design process, series of prototypes are typically released and enhancements from one system to the next are made by collecting a corpus of dialogues between users and prototypes to evaluate their performance. One way to avoid prototype releases is to use WOZ techniques of course but human users are still. Thus, another application of dialogue simulation by computer means is the evaluation of SDS like discussed in section II.4.5. Most powerful simulation systems for evaluation purpose are probabilistic. Of course, a first way to use simulation to evaluate SDS performance is to build a handcrafted system able to interact with the SDS by some means and to evaluate the quality of the resulting dialogue sessions by using objective dialogue metrics [Lin & Lee, 2001]. While this technique avoids human involvement, it is driven by a set of fixed parameters and performs always the same in a given situation. Parameters have to be modified in order to obtain different behaviours. Another use of a simulation system for SDS evaluation is the expansion of a data set. Indeed, if user evaluations of one or more prototypes have already been realised, the designers have a data corpus at their disposal that is hopefully representative of possible interactions between users and prototypes. This corpus can then be used to create a generative model of the SDS's environment [Eckert *et al*, 1998]. The model is then theoretically able to produce new dialogues having the same statistical properties than those in the corpus and enables to evaluate unseen dialogues. This way, designers can obtain a more accurate evaluation of the system.

Finally, the last simulation application discussed here is the simulation for strategy learning purposes. Indeed, once a way to simulate as many dialogues as desired has been obtained as well as a way to evaluate the resulting dialogues, it is natural to think about exploiting this framework to learn optimal strategies from experiences. The evaluation method is then used to build an optimality criterion. As the learning process requires lots of dialogues to converge toward an optimal strategy, computer-based simulation is very valuable. One of the first attempts is described in [Levin & Pieraccini, 1997] although the idea of has been firstly suggested in [Walker, 1993]. More about dialogue strategy learning will be discussed in the Third Part of this text. Nevertheless, the dialogue simulation part is motivated by the potentialities of simulation in the purpose of strategy learning by RL means. In this framework, real interactions between a learning agent and human users would be unfeasible since a large number of dialogues are needed, some of them can be long and seem silly and the overall operation would be very expensive and time consuming.

IV.2. Probabilistic Simulation Environment

As seen in section II.3, most of low-level components of an SDS make heavy use of probabilistic methods. Unlikely, for a long time, the DM design has been based on heuristics and iterative refinements. It was thus reasonable to attempt at applying probabilistic techniques to the design and simulation of the DM too. Indeed, probabilistic techniques offer a powerful theoretical framework and let foresee learning capabilities.

With this idea in mind and defining a DM in terms of internal states and actions within the framework of MDPs, the following general black box scheme for the environment of a DM can be drawn.

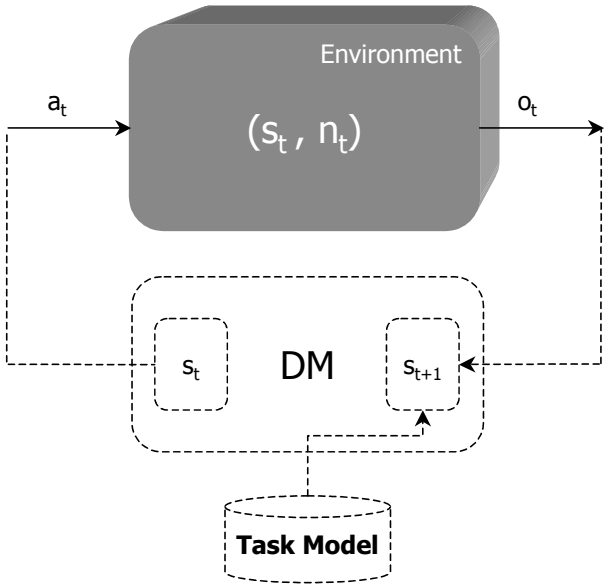


Fig. 22: Black box scheme of a simulation environment

In this figure (where dotted objects are not part of the environment),

- t is a discrete representation of time snapped at each dialogue turn,
- a_t is the action performed by the DM and transmitted to the environment at time t,
- s_t is the internal state of the environment at time t (it is closely related to the internal state of the DM),
- n_t is the contribution of noise at time t (including real acoustic noise but also modelling other errors occurring in the lower level processes),
- o_t is the observation the DM can perceive at time t after its action a_t has been processed by the environment and is typically a sequence of concepts,
- s_{t+1} is the internal state at time t+1.

In a first probabilistic approach of the simulation problem, let's express the probability of a given configuration as the joint probability of action, observation at time t and state at time t+1 given the internal state at time t and the noise:

Erreur ! Des objets ne peuvent pas être créés à partir des codes de champs de mise en forme. (IV.1)

Several assumptions can be made without loss of generalities. For example, it can be reasonably assumed that the task model (that interprets the observation in order to build a new internal state for the DM) is independent from the action and the noise:

$$P(s_{t+1} \mid o_t, a_t, s_t, n_t) = P(s_{t+1} \mid o_t, s_t) \tag{IV.2}$$

It is also reasonable to think that the DM does not take the noise into account in order to decide which action to perform next, at least not directly. If it does, it is through the observations and thus, according to the state representation extracted from it:

$$P(a_t \mid s_t, n_t) = P(a_t \mid s_t) \tag{IV.3}$$

Thus:

$$P(s_{t+1}, o_t, a_t \mid s_t, n_t) = \underbrace{P(s_{t+1} \mid o_t, s_t)}_{\text{Task Model}} \cdot \underbrace{P(o_t \mid a_t, s_t, n_t)}_{\text{Simulation Env.}} \cdot \underbrace{P(a_t \mid s_t)}_{\text{DM}} \tag{IV.4}$$

All the problem of dialogue simulation is to evaluate the second term of this last expression, while the evaluation of the last term depends on the actual strategy of the DM.

IV.2.1. Several Probabilistic Proposals for Simulating the Environment

As said previously, there have been several attempts to dialogue simulation before. This section first describes previous work in the field and finally gives a first description of a more realistic proposal.

State Transition Model

This model is not the first that appeared in the literature but it seems to be the simplest. Indeed, in [Walker, 2000] the author describes a simulation system based on the extraction of transition probabilities from a corpus of dialogues obtained by direct utilisation of a prototype by users. Other descriptions and uses of this model can be found in [Singh *et al*, 1999]. In this framework, both the task model and the simulation environment terms are estimated by one transition probability. Indeed, assuming that the outputs of the simulated environment are directly state representations ($s_{t+1} = o_t$) the following can be written:

$$\begin{aligned} \underbrace{P(s_{t+1} \mid o_t, s_t)}_{\text{Task Model}} \cdot \underbrace{P(o_t \mid a_t, s_t, n_t)}_{\text{Simulation Env.}} &= P(s_{t+1}, o_t \mid a_t, s_t, n_t)_{s_{t+1}=o_t} \\ &= P(s_{t+1} \mid a_t, s_t, n_t) \end{aligned} \tag{IV.5}$$

In the abovementioned researches, the noise was also neglected, as it is quite difficult to include in this framework. Thus, the estimated transition probability was expressed as:

$$P(s_{t+1} \mid a_t, s_t, n_t) \approx P(s_{t+1} \mid a_t, s_t) = \mathcal{T}^{a}_{ss'} \tag{IV.6}$$

The estimated probability is actually the transition probability of the underlying dialogue's MDP representation (see section III.2.1) and no task model interpretation is needed anymore but a dialogue corpus.

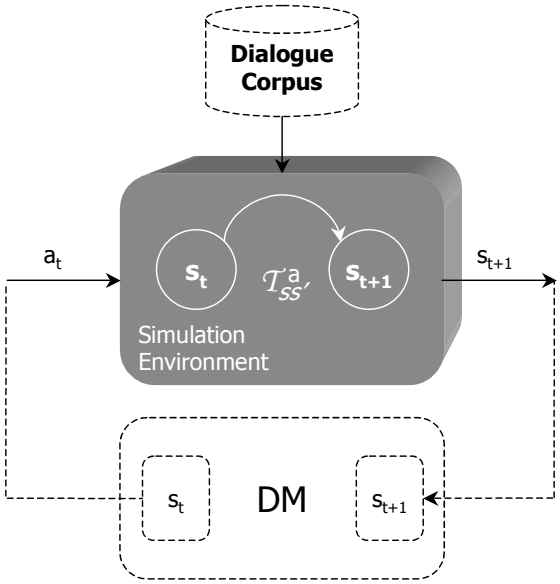


Fig. 23: State-transition simulation model

The principal drawback of this model (besides the need of data collection) is that the dialogue's MDP representation must be known in order to collect suitable data for probabilities estimation. The model can then be used for one pair of state space and action set but it is hardly extendable to new configurations, except if a new state space (or action set) can be derived from the previous one. Moreover, the entire state space must be visited a sufficient number of times in order to have reliable estimates of the probabilities, which is quite hard if not impossible. Finally, these estimates are obtained by following a particular strategy, which can make the model poorly reliable. Nevertheless, it has proved to be useable for evaluation purposes.

Simulating Individual Components

According to the prior knowledge of what composes the DM's environment, it is natural to include stochastic models of each component in the simulation environment. Remember that the DM's environment contains all the SDS components except the DM itself and the WK but also the user. It is precisely this last one that has been modelled first in [Eckert *et al*, 1997]. At the beginning, the authors described their simulated user as a probabilistic model based on the interaction history with the probability of user pronouncing utterance u_t at time t according to the history being

$$P(u_t \mid sys_t, u_{t-1}, sys_{t-1} \ldots), \tag{IV.7}$$

where sys_t stands for the utterance pronounced by the system at time t. Rapidly, they concluded that only a bigram model would be tractable and the context has been reduced to the previous system's utterance. The probability of the user pronouncing utterance u_t at time t according to the history became:

$$P(u_t \mid sys_t) \tag{IV.8}$$

The utterance u_t can be assimilated to observation o_t while the system's utterance sys_t can be assimilated to the action a_t. Thus, the last expression means that the simulated user is memory-less and that it only answers to system's questions even if it already answered before or if it didn't previously provide necessary information to answer the question. The current state is not taken into account, the user is not goal directed and no error modelling was included (noise is not taken into account neither):

$$P(o_t \mid a_t, s_t, n_t) \approx P(o_t \mid a_t) = P(u_t \mid sys_t) \tag{IV.9}$$

Moreover, those probabilities have to be extracted from data corpus. As the problem of data collection is recurrent, the same authors, who faced the problem, proposed another set of probabilities (still based on the bigram model) for describing the user's behaviour that can be handcrafted by experts [Levin *et al*, 2000].

A goal-directed simulated user has been introduced in [Scheffler & Young, 1999]. The model follows a quite complex 'grammar-based' scheme and its design is very task-dependent and also requires a corpus. Nevertheless, this model takes the current state into account.

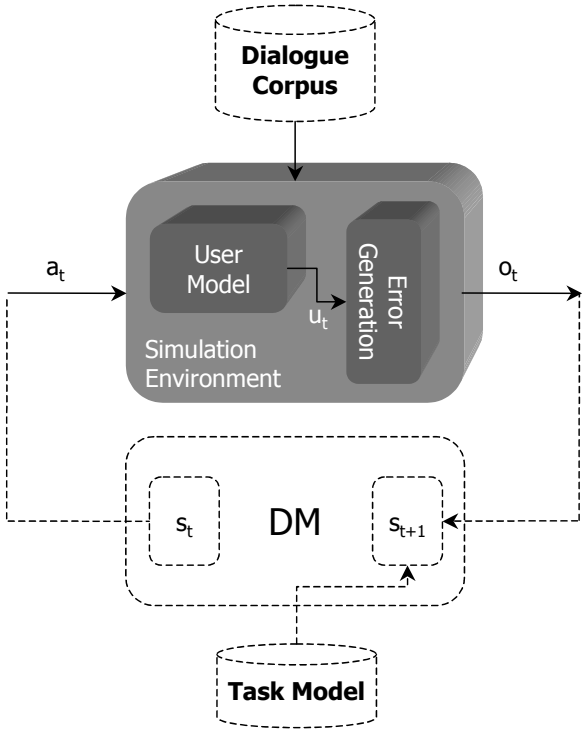


Fig. 24: Simulation environment with individual components

On another hand, several attempts to model errors occurring when acquiring users intentions have been realised. Most of studies rely on the well-known problem of ASR errors (insertion, deletion and substitution of words in a recognized sequence). Once again, a probabilistic model is built to simulate the error generation process. The model is included in the environment and affects the produced observation (Fig. 24). Until now, very simple error modelling has been realised (except in [Scheffler & Young, 1999] were a quite complex error model is described). According to this model, the next expression is derived:

$$P(s_{t+1}, o_t, u_t, a_t \mid s_t) = \underbrace{P(u_t \mid a_t, s_t)}_{\text{User}} \cdot \underbrace{P(o_t \mid u_t, a_t, s_t)}_{\text{Error Generation}} \cdot \underbrace{P(s_{t+1} \mid o_t, u_t, a_t, s_t)}_{\text{Task Model}} \cdot \underbrace{P(a_t \mid s_t)}_{\text{DM}} \quad \text{(IV.10)}$$

With the same assumptions than previously and assuming that the user's utterance actually depends on the current state (at the opposite of the memory-less model described in the above), equation IV.10 becomes:

$$P(s_{t+1}, o_t, u_t, a_t \mid s_t) = \overbrace{\underbrace{P(u_t \mid a_t, s_t)}_{\text{User}} \cdot \underbrace{P(o_t \mid u_t, a_t, s_t)}_{\text{Error Generation}}}^{\text{Simulation Env.}} \cdot \underbrace{P(s_{t+1} \mid o_t, s_t)}_{\text{Task Model}} \cdot \underbrace{P(a_t \mid s_t)}_{\text{DM}} \quad \text{(IV.11)}$$

Notice that this model doesn't take the noise into account like most of models described in the literature. The simulation problem is then to evaluate the two first terms of this last expression, which is generally done by collecting a data corpus like in [Scheffler & Young, 1999].

Yet, an even more corpus-based technique has been proposed in [López-Cózar *et al*, 2003] where real ASR and NLU systems are used and the simulated data come from a corpus of recorded utterances that are selected according to a scenario itself coming from a corpus. The major drawback of this proposal is that it can only be used with a completely implemented and working SDS, and is therefore useful only for evaluation.

Toward A More Realistic Simulation Environment

In order to build a more realistic simulation environment for SDS evaluation, several facts should be taken into account. Here are some reflections that can be done about the problem:

1. The simulation environment should be the less task-dependent as possible.
2. The simulated user should be goal-directed and should behave consistently according to its goal.
3. A human user would have a sufficient knowledge about the task to have a consistent goal according to the task.
4. The DM's action set includes actions that are meant to become synthesized speech utterances spoken to the user but also actions that are meant to query or modify the WK.
5. If there can be errors introduced by the input processing systems of the SDS, outputs generation can also be error-prone.
6. If the simulation environment is meant to allow SDS evaluation, it should provide information about metrics that affect user's satisfaction.
7. Noise influences the inputs processing results but also the way synthesized speech is perceived.
8. Synthesized speech quality and the length of system's utterances influences user's satisfaction [Walker *et al*, 2001].

According to this list the simulation environment on Fig. 25 is proposed as an extension of the environment described in [Pietquin & Renals, 2002] and [Pietquin & Dutoit, 2002]. The inclusion of the WK in the environment answers to statements 1, 2, 3 and 4 of the above list. Indeed, if the goal is to obtain a user model as task-independent as possible but to make it able to build a task-dependent goal at the beginning of each simulated dialogue session, it should access to the task model included in the WK.

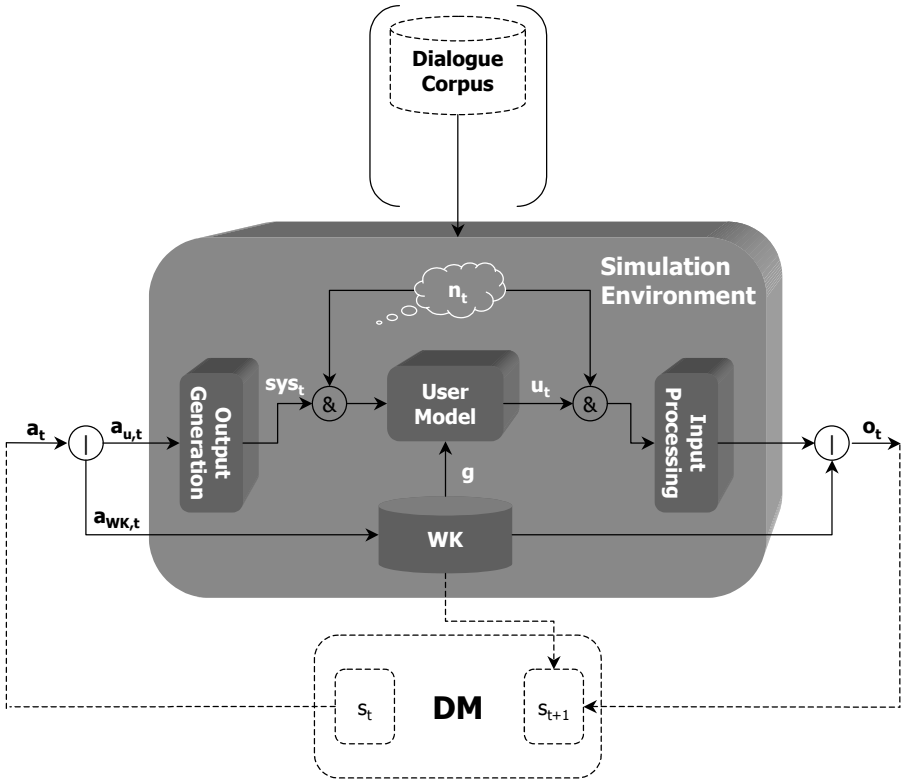


Fig. 25: A more realistic simulation environment

Moreover, if DM's actions can affect the WK, it should definitely be considered as a part of the environment. In agreement with statement 5, a model of the output generation blocks has been included to the simulation environment. This model also addresses problems raised by statements 6 and 8. A noise source has been added and generated noise is mixed with SDS generated output before it reaches the user model and with user model's outputs in conformity with statement 7. Finally, the DM's action set A has been split into two parts (A_u and A_{WK}) in order to address problems expressed by statement 4:

$$\underbrace{\{a_i\}_{i=1\to n}}_{A} = \underbrace{\{a_{u,j}\}_{j=1\to l}}_{A_u} \cup \underbrace{\{a_{WK,k}\}_{k=1\to n-l}}_{A_{WK}} \qquad \text{(IV.12)}$$

In Fig. 25, variables are:

- s_t is the internal state at time t,
- a_t is the action performed by the DM and transmitted to the environment at time t,
- $a_{u,t}$ is the action meant to become synthesized speech utterances spoken to the user (null if $a_t \in A_{WK}$) at time t,

- $a_{WK,t}$ is the action meant to access or modify the WK (null if $a_t \in A_u$) at time t,
- sys_t is the system's utterance, derived from action $a_{u,t}$, by the output generation block at time t,
- u_t is the concept sequence produced by the user model at time t in response to the concepts he could retrieve from sys_t (we do not work at the word level here as it is intractable and useless to model all the manner a user can express a set of context),
- g is the user's goal,
- n_t is the contribution of noise at time t,
- o_t is the observation the DM can perceive at time t after its action a_t has been processed by the environment,
- s_{t+1} is the internal state at time t+1.

According to these definitions, the joint probability is given by:

$$\begin{aligned} P(s_{t+1},o_t,a_t,u_t,sys_t,g \mid s_t,n_t) = {} & P(s_{t+1},o_t,a_t \in A_u,u_t,sys_t,g \mid s_t,n_t) \\ & + P(s_{t+1},o_t,a_t \in A_{WK},u_t,sys_t,g \mid s_t,n_t) \end{aligned} \quad \text{(IV.13)}$$

With the same assumptions than before and keeping in mind that actions on the WK give deterministic results, the second term of this expression can be expressed as:

$$\begin{aligned} P(s_{t+1},o_t,a_t \in A_{WK},u_t,sys_t,g \mid s_t,n_t) &= P(s_{t+1} \mid o_t,s_t,a_{WK,t},n_t) \cdot \\ &\quad \cdot \overbrace{P(o_t \mid a_{WK,t},s_t,n_t)}^{\text{Deterministic}} \cdot P(a_{WK,t} \mid s_t,n_t) \\ &= \underbrace{P(s_{t+1} \mid o_t,s_t)}_{\text{Task Model}} \cdot \underbrace{P(a_{WK,t} \mid s_t)}_{\text{DM}} \end{aligned} \quad \text{(IV.14)}$$

Thus, as long that the WK is part of the simulation environment, nothing has to be modelled in this term. The first term of the same expression can be decomposed as:

$$\begin{aligned} P(s_{t+1},o_t,a_t \in A_u,u_t,sys_t,g \mid s_t,n_t) = {} & \underbrace{P(sys_t \mid a_{u,t},s_t,n_t)}_{\text{Outputs Generation}} \\ & \cdot \underbrace{P(g \mid sys_t,a_{u,t},s_t,n_t)}_{\text{Goal Modification}} \\ & \cdot \underbrace{P(u_t \mid g,sys_t,a_{u,t},s_t,n_t)}_{\text{User}} \\ & \cdot \underbrace{P(o_t \mid u_t,g,sys_t,a_{u,t},s_t,n_t)}_{\text{Inputs Processing}} \\ & \cdot \underbrace{P(s_{t+1} \mid o_t,u_t,g,sys_t,a_{u,t},s_t,n_t)}_{\text{Task Model}} \\ & \cdot \underbrace{P(a_{u,t} \mid s_t,n_t)}_{\text{DM}} \end{aligned} \quad \text{(IV.15)}$$

In addition to previous assumptions, the following can also been made:

- The output generation is independent from the noise and the current state. Indeed, the NLG subsystem generates a text according to concepts embedded in the action only. This text is then synthesized

without taking noise into account although one could think about output signal level adaptation. This doesn't mean that the noise doesn't affect the perception of the system's utterance by the user but the production process is not affected by the noise.

- The user model doesn't change its goal during a dialogue session and all goals have the same probability to be chosen. Noise will be kept as a conditioning variable as it can affect the understanding of the system's utterance.
- User's utterance is independent from the actual DM's action. The action is transmitted to the user by the system's utterance sys_t.
- The inputs processing block results are independent from the user's goal and the system's utterance. Indeed, the system cannot be aware of the user's goal to tune its subsystems. Moreover, if the actual DM action can be responsible for some tuning, the spoken realisation of the concepts embedded in the action is not responsible for any tuning.
- The task model is independent from the user's utterance and goal and from the system's utterance and noise.

The previous expression can therefore be rewritten:

$$\begin{aligned} P(s_{t+1}, o_t, a_t \in A_u, u_t, sys_t, g \mid s_t, n_t) = & \underbrace{P(sys_t \mid a_{u,t})}_{\text{Outputs Generation}} \cdot \underbrace{P(u_t \mid g, sys_t, s_t, n_t) \cdot P(g)}_{\text{User}} \\ & \cdot \underbrace{P(o_t \mid u_t, a_{u,t}, s_t, n_t)}_{\text{Inputs Processing}} \\ & \cdot \underbrace{P(s_{t+1} \mid o_t, a_{u,t}, s_t)}_{\text{Task Model}} \cdot \underbrace{P(a_{u,t} \mid s_t)}_{\text{DM}} \end{aligned} \quad \text{(IV.16)}$$

The simulation problem can be summarised as the problem of evaluating the four first terms of this expression:

$$\underbrace{P(sys_t \mid a_{u,t})}_{\text{Outputs Generation}}, \underbrace{P(u_t \mid g, sys_t, s_t, n_t), P(g)}_{\text{User}}, \underbrace{P(o_t \mid u_t, a_{u,t}, s_t, n_t)}_{\text{Inputs Processing}} \quad \text{(IV.17)}$$

As equal probabilities for all goals are assumed, evaluating P(g) is quite trivial but the evaluation of other terms will be the subject of the following chapters.

On another hand, the simulation environment should provide evaluation indices for each dialogue session. All the components of the environment should then provide metrics indicating how well they performed their job. An additional block dedicated to the computation of an instantaneous evaluation c_{t+1} of the turn is then included in the simulation environment (Fig. 26). Inputs of this block are metrics supplied at time t by each component and are referred to as follow on the figure:

- $\{c_t^{out}\}$ are the metrics provided by the output generation blocks
- $\{c_t^{in}\}$ are the metrics provided by the input processing blocks
- $\{c_t^{wk}\}$ are metrics provided when accessing the WK
- U^S is the overall user's satisfaction which is typically provided once, at the end of each dialogue session.

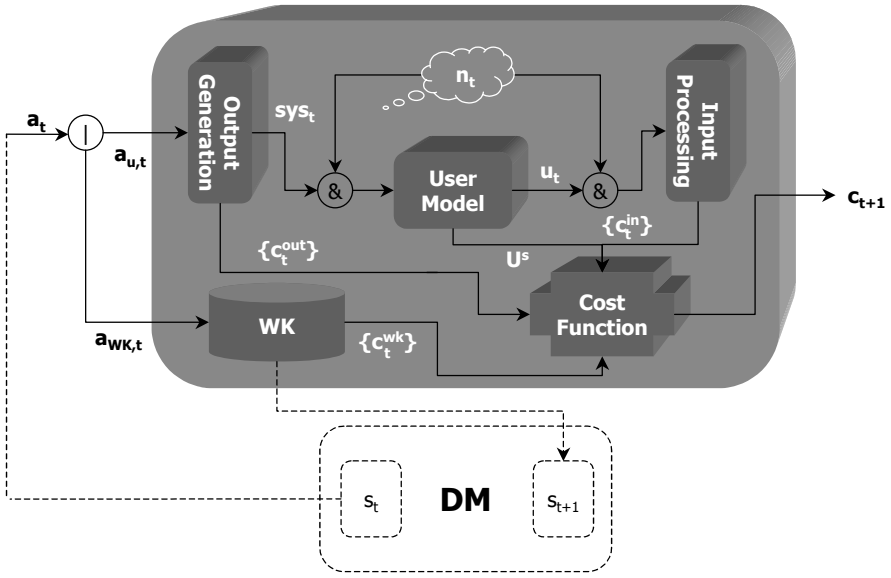


Fig. 26: Simulation environment with cost function block

IV.2.2. Implementation Details

Each of the simulation environment components will be discussed in the two following chapters. Nevertheless, it is important to define more properly how the communication takes place between components. Therefore, the structure of internal variables should be detailed.

To do so, one central point to be considered is the task representation to adopt. Indeed, messages passed from one component to another often contain pieces of information about the task. In the next chapters, it will be assumed that the AVM task representation (see II.3.5) is chosen for the following reasons, among others:

- the AVM representation is suitable for a large set of dialogue applications including those discussed in this thesis (such as form filling and information retrieval applications),
- it is compatible with relational database representation of the task (which is often a starting point to the deployment of voice-enabled interfaces),
- it is convenient for factored representation of messages.

This Attribute-Value (AV) data representation will be kept for most of internal messages. Consequently, a certain number of internal variables have a partially imposed structure:

- the user's goal (g) is also represented as an AVM extracted from the task representation,
- the user model's output is a set of AV pairs,
- what the user model processes is not a complete utterance synthesized by the output processing block mixed with noise but also a set of AV pairs.

Actually, these statements can be interpreted by saying that in the simulation environment, communication takes place at the intention level rather than at the word sequence or speech signal level, as it would be in the real world. The user's goal is then a collection of all the intentions the user has to communicate to the system. An intention is regarded as the minimal unit of information that a dialogue participant can express independently. Intentions are closely related to concepts, speech acts or dialogue acts.

It is unnecessary to model environment's behaviour at a lower level, because from the point of view of the DM, only the high level communication is useful (see Fig. 3). Additionally, intention-based communication allows error modelling of all the parts of the system. More pragmatically, it is simpler to automatically generate concepts compared with word sequences (and certainly speech signals), as a large number of utterances can express the same intention (and above all for user's utterances).

Nevertheless, there is another way to interpret the choice of intention-based communication. For instance, it is reasonable to say that a user is able to translate a speech signal synthesized by the system in a set of AV pairs. It is then realistic to model the user by an intention-processing module.

Chapter V:
User Model

V.1. About User Modelling

User modelling is a quite recent field of AI researches and is still a wide-open problem. In general, a User Model (UM) is used in the aim of adapting a system's behaviour to what its user is expected to do and to his/her needs and preferences. It is therefore mainly used for user's knowledge and goal inference [Zukerman & Albrecht, 2001]. For example, the MS Office Assistant uses a UM to infer user's needs and knowledge through natural language entries [Horvitz *et al*, 1998]. User models are also used in games in order to obtain team behaviour between the user and his/her virtual squad or at the opposite to fight the user [Suryadi & Gmytrasiewicz, 1999].

User modelling for interaction adaptation commonly accepts two different approaches. Historically, the user modelling has first focused on the human *emulation* approach in which the UM is used in order to create 'human-like' behaviour for the system [Kass & Finin, 1988]. Actually, this 'human-like' behaviour is not as desirable as imagined since human users generally behave differently when talking to a machine [Jönsson & Dahlbäck, 1988]. Thus, the *complementary* approach that exploits this asymmetry between human and computer to build new interaction and collaborative possibilities seems preferable [Fisher, 2001].

Here, the purpose of user modelling is not only to model user's knowledge and to infer user's goal for use in a working system but to predict user's behaviour given a SDS action in order to simulate dialogues (more like a generative model). In this purpose, several approach ca also be considered.

In the case no data is available about users of a given system, which is generally the case in the early stage of the design process, a method has been proposed in the early ages of user modelling: *stereotyping* [Rich, 1979]. Stereotyping is a very simple concept seeming obvious but it has been widely used and after all, is still used in general. A stereotype captures default information about groups of users and it is assumes that a user belonging to a group acts in a roughly same way than the others of the group. The design of stereotype-based models implies the judicious choice of user groups and key characteristics. Initially, stereotypes were based on people's description of themselves.

In the case data are available or online learning is feasible, it is possible to learn parameters of a UM. Anyway, stereotyping method can serve as a starting point of the

learning process. Whatever the framework used for modelling the user, two learning methods can be considered: *content-based* and *collaborative* learning [Zukerman & Albrecht, 2001]. Collaborative learning models are very similar to stereotype-based models in such a way that it is assumed that that user behaves in a similar way of other 'like-minded' users (those belonging to the same group). Parameters are learned from a data set collected on a user population that is supposed to represent correctly a group. Content-based learning is used whenever it can be assumed that the past behaviour of a given user is a reliable indicator of her/his future behaviour. This method learns models of each system's user, which means that it is only applicable for systems often used by the same users (like an software for personal computers, for instance).

Finally, a last distinction has to be made between *rule-based* and *statistical* models. The former are often handcrafted and allows for little uncertainty management while the later is more suitable for learning and allows more flexible behaviour. They are widely used nowadays and these models will be on focus.

Statistical user modelling in SDS has a non-empty history but employing UMs for the purpose of dialogue simulation is not so common. Yet, two recent examples can be found in [Eckert *et al*, 1997] and [Scheffler & Young, 1999]. In [Eckert *et al*, 1997], the authors developed a very simple statistical model in which the user's behaviour is independent of the history of the interaction and the user's goal. This strong assumption simplifies greatly the UM and the parameters to be learned. In [Scheffler & Young, 1999], the authors developed a quite complex goal-directed UM based on statistical grammars capturing dialogue context but the parameters to be learned are very task-dependent. It requires the use of an accurate prototype in order to estimate parameters and a lot of expertise to be built.

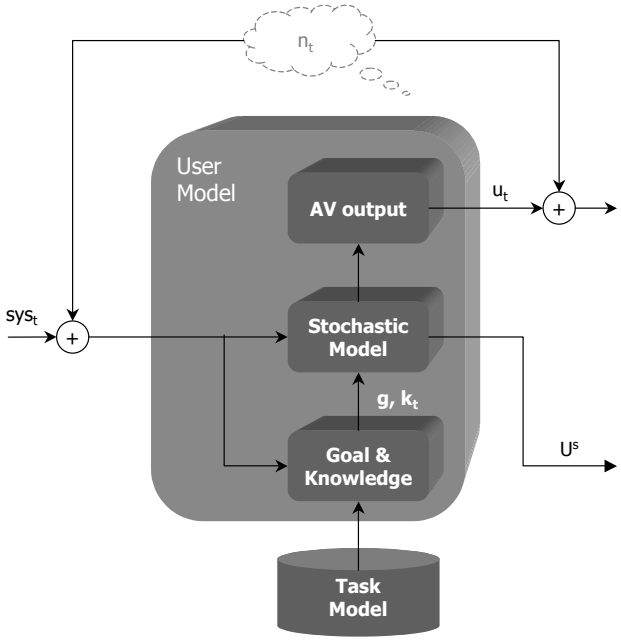


Fig. 27: Stochastic User Model

Fig. 27 shows the details of the UM. It is composed of three blocks. The lower one represents the user's knowledge (k_t) at time t, which is updated by the system's utterance at time t and of which the user's goal (g) is a subset. The stochastic model is the main part of the UM and implements UM's behaviour. It takes the user's knowledge and goal and the system's utterance as inputs. Readers should remember that the system's utterance is

considered to be a set of AV pairs where the attributes will be denoted $\{s^\beta\}$ and the set of possible values for attribute s^β will be denoted $V^\beta = \{ v_i^\beta \}$. The upper block transforms output of the stochastic model in a set of AV pairs transmitted to the ASR model in which attributes bellows to $\{u^\alpha\}$ and the set of possible values for u^α is $V^\alpha = \{ v_i^\alpha \}$. Remember that according to equation IV.17, the user model should simulate user's behaviour by providing an utterance u_t according to $P(u_t \mid g, sys_t, s_t, n_t) \cdot P(g)$.

V.2. A First Simple Model

In a first attempt to realise a model of a SDS user, the theory developed in [Eckert *et al*, 1997] was used as a starting point since it is more likely to be obtainable by automated means. In this paper, the authors make the assumption that the user's utterance at time t is conditioned by the history of the dialogue session:

$$P(u_t \mid sys_t, u_{t-1}, sys_{t-1} ..., u_0, sys_0) \tag{V.1}$$

These probabilities being too difficult to obtain whatever the method (even by using a corpus of data, it is unrealistic to think that the same history will appear a sufficient number of time in order to compute reliable value of probabilities for all user's utterances at time t), the authors make then the very strong assumption that the user's utterance is only conditioned by the system's previous utterance:

$$P(u_t \mid sys_t) \tag{V.2}$$

Once again, the authors didn't have a sufficiently large amount of data in order to compute those probabilities for a mixed-initiative SDS (they worked on the ATIS corpus which is only compose of answer to an 'how may I help you' prompt). Thus in [Levin *et al*, 2000] they defined a set of simpler probabilities and handcrafted most of them:

- Probabilities associated with response to greeting:
  - P(n|Greeting) is the probability to provide n AV pairs in a single utterance when the greeting message is prompted.
  - P(*A*) is the probability distribution of attributes
  - P(*V*|*A*) is the probability distribution of values for each attribute in *A*.
- Probabilities associated with response to constraining questions:
  - $P(u^\alpha|s^\beta)$ is the probability of the user to provide a value for attribute u^α when prompted for attribute s^β.
  - $P(n|s^\beta)$ is the probability of the user to provide n unsolicited AV pairs when prompted for attribute s^β. The unsolicited attributes are provided according to the distribution P(*A*), like for the answer to greeting prompt.
- Probabilities associated with response to relaxation prompts:
  - $P(yes|s^\beta) = 1 - P(no|s^\beta)$ is the probability of the user to accept the relaxation of attribute s^β.

This set of probabilities is quite simple and can be handcrafted if necessary. It has been used in a first attempt of dialogue simulation for strategy learning. Nevertheless, this UM suffers of a lack of consistency for two reasons:

- If the UM is prompted twice for a same attribute, it can provide two different values since the probabilities are not conditioned on a particular goal (this UM is not goal-directed)
- If the UM is prompted infinitely for the same attribute, it will answer infinitely (and maybe change its answer) since the UM doesn't 'remember' that the sentence has already been prompted.

These two problems can lead to unrealistic simulated dialogues but also to infinite dialogues since if the user's changes its answer to a same question infinitely, the task might not be possible to achieve. Moreover, this UM cannot express is (un)satisfaction about the dialogue and close the session for example when it is not satisfying.

For instance, suppose an SDS dedicated to train ticket booking which possible actions A_S are greeting prompt (greeting), constraining questions (const), confirmations (conf), relaxation requests (rel) or closing the dialogue (close). Attributes of those actions are the departure city (dep), destination city (dest) and the travelling class (class). The strategy followed by the SDS can be summarized as follow:

- starting with the 'greeting' action
- ask for missing attributes until they are all known
- confirm before selling the ticket
- ask for relaxation when no ticket can be sold
- close when all has been confirmed or after a fixed number of turns.

	Intentions	Expanded Intentions
sys_0	A_S = greeting	*Hello! How may I help you?*
u_0	u^1 = dest v^1 = 'Namur'	I'd like to go to Namur.
sys_1	A_S = const(dep)	*What is your departure City?*
u_1	u^1 = dep v^1 = 'Mons'	I'd like to start from Mons.
sys_2	A_S = const(class)	*Would you like a business or an economy class ticket?*
u_2	u^1 = class v^1 = 'Business'	Business class, please.
sys_3	A_S = conf(all)	*Ok, please confirm the following request: you want a business class ticket to go from Mons to Namur?*
u_3	u^1 = conf v^1 = *false* u^2 = dep v^2 = 'Namur'	No, I want to start from Namur.
sys_4	A_S = conf(dep, dest)	*You want go from Namur to Namur?*
u_4	u^1 = conf v^1 = *true*	Yes.
sys_5	A_S = rel(dest)	*It is impossible to go from Namur to Namur. Do you want to choose another destination?*
…		…

Table 1: An example of dialogue with a memory-less UM

This UM has been implemented and tested on a simulated implementation of the SDS described in the above. We suppose that the communication is error-free. A typical

problematic dialogue is depicted in Table 1 (exchanged intentions has been expanded into a word sequence representation in the right column for comprehensive purpose). We can say from this little example that an indication about the UM performance is the mean length of a dialogue session. Since the SDS itself decides to close the dialogue when the number of turns crosses a given threshold, the value for the dialogue length (in number of turns) can vary from the minimum number of turns allowed by the strategy to this threshold. The number of turns is minimum when the UM provides the three attributes and their values when answering the greeting, confirm them and the ticket is available (that is when the value for dest is different of the value for dep). Thus, the minimum number of turns is 3 (greeting, conf, close).

Notice also that in this case, the mean number of turns will be dependent of the probability of obtaining the same value for dest and dep:

$$P\left(v^{dest} = v_i^{dest}, v^{dep} = v_i^{dest}\right) = P\left(v^{dest} = v_i^{dest}\right) \cdot P\left(v^{dep} = v_i^{dest}\right) \quad (V.3)$$

This effect is shown on Fig. 28. It is of course an undesired effect to avoid since the evaluation of the SDS is made more difficult.

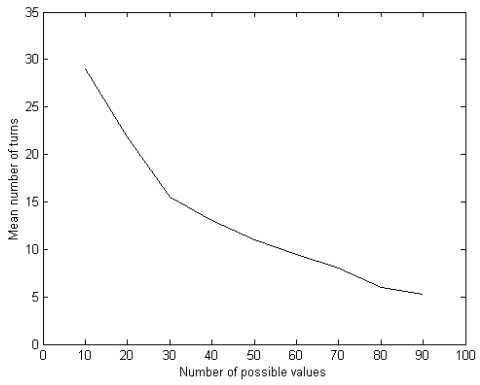


Fig. 28: Average number of turns vs. number of possible values for arguments

V.2.1. A Goal-Directed User Model with Memory

To overcome problems described in the above, a similar UM is proposed but each probability is then also conditioned on the user's goal and on a small amount of information about the history of the interaction. The UM has then to build a new goal at the beginning of each dialogue session and will act consistently according this goal all along the interaction. Moreover, the information kept by the UM about the history mainly counts for occurrence of the same system's utterance and a new probability is added: the probability to hang off.

According to the notation of Fig. 27, the information kept by the UM about the history is roughly the user's knowledge at time t (k_t) and the goal is denoted g. The goal contains a set of AV pairs that will condition the answers of the UM and the knowledge contains few information about the history of the current dialogue (see Table 2).

The goal not only contains AV pairs but also priorities for each AV pair that will be helpful when prompted for relaxation of some attribute. An AV pair with high priority will be less likely to be relaxed. The knowledge of the UM is essentially composed of

counters initialised at 0 and incremented each time a value is provided for a given attribute. This will enable the UM to supply new AV pairs when mixed-initiative behaviour is provided (and not to provide already supplied values) and to react to unsatisfactory behaviour of the system.

Goal			Know.
Att.	Val.	Prior.	Count
g^1	v_i^1	p^1	k^1
g^2	v_j^2	p^2	k^2
⋮	⋮	⋮	⋮
g^n	v_m^n	p^n	k^n

Table 2: UM's goal and knowledge representation

The probabilities then become:

- Probabilities associated with responses to greeting:
  - P(n|Greeting, g) is the probability to provide n AV pairs in a single utterance when the greeting message is prompted according to the goal g.
  - P(A|k_t, g) is the prior probability distribution of attributes given the knowledge at time t of the UM about the history and its goal. It is modified by the knowledge since already provided attributes ($k^{\alpha} > 0$) see their probability decreasing.
  - P(V|A, g) is the probability distribution of values of each attribute in A given the goal. The probability of providing values appearing in the goal is maximal while assessing non-null values to other probabilities can simulate lack of cooperativeness.
- Probabilities associated with responses to constraining questions:
  - $P(u^{\alpha}| s^{\beta}, k_t, g)$ is the probability of the user to provide a value for attribute u^{α} when prompted for attribute s^{β}. Here again the knowledge and the goal will determine which value will be provided.
  - $P(n| s^{\beta})$ is the probability of the user to provide n unsolicited AV pairs when prompted for attribute s^{β}. Attribute will be provided according to P(A|k_t, g) which means that the probability of providing attributes that have not yet been provided can be set to higher values since the knowledge of the user conditions the probability.
- Probabilities associated with responses to relaxation prompts:
  - $P(yes| s^{\beta}, k_t, g) = 1 - P(no| s^{\beta}, k_t, g)$ is the probability of the user to accept the relaxation of attribute s^{β}. Priorities associated to AV pairs in the user's goal will modify the probability of certain attributes to be relaxed when asked for.
- Probabilities associated to user satisfaction:

- P(close| s^β, k_t, g) is the probability to close the dialogue session before the natural end when asked for the argument s^β because of unsatisfactory behaviour of the system, for example when the counter of k^β associated to the attribute s^β is over a given threshold.

At the opposite of the previous described model, the mean length of a dialogue is now a function of user's goal (and thus of the task structure), knowledge and degree of cooperativeness, but not of the number of possible values that can be taken by each of the arguments. The model can be learned or handcrafted. Indeed, even if the number of probabilities is larger than in the previous model, they can be assessed *a priori* by using a set of rules if no data is available:

- The probabilities P(u^α| s^β, k_t, g) is set to a high value for $u^\alpha = s^\beta$ and non-null values can be assigned to other values to simulate the lack of cooperativeness.
- P(A|k_t, g) is conditioned on the knowledge and already provided attributes ($k^\alpha > 0$) have lower probabilities. Moreover, the priority of an attribute increases it probability to be provided.
- P(V|A, g) is maximal for the value appearing in the goal for the attribute while assessing non-null values to other probabilities can simulate lack of cooperativeness (a special care must be take to ensure that the sum of the probabilities is equal to 1).
- P(yes| s^β, k_t, g) is assessed according to the priority of the attribute: the higher the priority, the lower the probability to relax it.
- The probability P(close| s^β, k_t, g) to close the dialogue when asked for the attribute s^β is based on a threshold for k^β.

The generated dialogues are consistent according to the goal and infinite dialogues are impossible if the probability P(close| s^β, k_t, g) is non null for all values of s^β.

Moreover, the specification of the goal as a set of AV pairs enables the measure of the task completion. For example, in the case of a form filling dialogue or a voiced-enabled interface for database querying, the system can access the AV pairs transmitted from the UM to itself and then compare them. The kappa coefficient described in section II.4.5 can be used for task completion measure, for instance.

The task completion measure can condition the user satisfaction, but unsatisfactory behaviour of the system, leading to a dialogue prematurely closed by the user can also lead to a decreased satisfaction. Thus, this model also allows a simple modelling of user satisfaction.

Application

Although this model is still very simple, it has been tested on a database querying system simulating an automatic computer dealing system like described in [Pietquin & Renals, 2002] and provided good results in the framework of strategy learning. In this experiment, the database contained 350 different computer configurations split into 2 tables (for notebooks and desktops) containing 5 fields each: pc_mac (pc or mac), processor_type , processor_speed, ram_size, hdd_size. To demonstrate this UM behaviour, it can be connected to a simple SDS that can perform 6 generic actions: greeting, constraining questions (const(arg)), confirmation (conf(arg)), relaxation queries

rel(arg), database queries (dbquery), close the dialogue (close). Each argument (arg) may be the table's type (notebook or desktop) or one of the 5 table fields.

In order to demonstrate the user's behaviour, a random SDS strategy is chosen. That is the system is only constrained to start with the greeting prompt and subsequently chose any other action with the same probability. Notice that the system did not enter into a negotiation phase in order to conclude the selling but retrieves a computer corresponding to the user's request in the database and provides information about the brand and the price of the selected product.

In this framework, the UM's goal can be described as in Table 3.

Goal			Know.
Att.	Val.	Prior.	Count
notebook_desktop	'desktop'	*high*	$k^1 = 0$
pc_mac	'PC'	*high*	$k^2 = 0$
processor_type	'Pentium'	*high*	$k^3 = 0$
processor_speed	800	*low*	$k^4 = 0$
ram_size	128	*low*	$k^5 = 0$
hdd_size	20	*low*	$k^6 = 0$

Table 3: UM's goal in the computer dealing task

The threshold for counts is set at 3 (that means that the after having provided the value for an attribute and confirmed it, the UM considers that this piece of information should not be asked again). Once again, the communication canal is supposed to be perfect and the understanding process as well. A typical problematic dialogue is shown in Table 4.

	Intentions	k	Expanded Intentions
sys_0	A_S = greeting		*Hello! How may I help you?*
u_0	u^1 = note_desk v^1 = 'desktop'	$k^1 = 1$	I'd like to buy a desktop computer.
sys_1	A_S = const(pc_mac)		*Would you prefer a PC or a Mac?*
u_1	u^1 = pc_mac v^1 = 'PC'	$k^2 = 1$	I'd prefer a PC.
sys_2	A_S = rel(note_desk)		*Don't you prefer a laptop?*
u_2	u^1 = rel v^1 = *false* (p = *high*)		No.
sys_3	A_S = conf(pc_mac)		*Can you confirm you want a PC?*
u_3	u^1 = conf v^1 = *true*	$k^2 = 2$	Yes, I want a PC
…	…	…	…
…	…	…	…
sys_i	A_S = const(pc_mac)		*Would you prefer a PC or a Mac?*
u_i	u^1 = close v^1 = *true*	$k^2 = 3$	Ok, bye! (*hang off*)

Table 4: A problematic dialogue with the goal-directed UM

This example illustrates two main points of the goal-directed model. First, a relaxation query is denied according to the goal priority of the attribute that was asked for relaxation. Second, the user closes the dialogue because the system behaved very badly. The update of the user's counts is also shown.

V.3. Bayesian Approach to User Modelling

The model described in the preceding section is based on a set of conditional probabilities. These probabilities are conditioned by the knowledge of the UM and by its goal. The goal representation and consequently the probability set is a function of the task structure. It has also been assumed that some intuitive rules can be used to assess *a priori* values for the probabilities while they could also be learned from a corpus of data obtained by releasing the SDS to a group of testing users.

After having enumerated all those features, it seems natural to implement a user model using the Bayesian Networks (BN) framework. Indeed, a BN graphically encodes conditional probabilities according to prior knowledge of the task and allows for parameterq and dependencies learning starting from prior estimates. Moreover, since it is a graphical model, it is also suitable for easy analysis and design by novices. Using BN for user modelling has already been realised in very few recent researches and mainly for goal inferring or knowledge representation like in [Horvitz *et al*, 1998] or in [Meng *et al*, 2000]. Here, it is proposed to use the BN framework for simulation purpose.

V.3.1. Network Structure

Designing a BN starts by the assessment of the topology and particularly by the enumeration of stochastic variables that will be represented by nodes in the network. In the case of the UM, variables are grouped in several sets:

- UM output variables:
  - The set of the k attributes transmitted by the UM's utterance u_t. This set will be denoted $U = \{u^\alpha\}_{\alpha=1}^{k} \subset A$.
  - The set of k values associated to transmitted attributes. This set will be denoted V.
  - A Boolean variable indicating whether the UM closes the dialogue session. It will be denoted 'close' = {0,1}.
- UM internal variables:
  - The set of knowledge variables. In the case of simple counts like in the preceding example, $K = \{k^\alpha\}$.
  - The set of goal variables including AV pairs contained in the UM goal and the optional priorities: $G = \{[a^\alpha, v_i^\alpha], p^\alpha\}$
- UM input variables (also system's output variables):
  - The type of system's action A_S. The allowed types can be constraining questions, relaxing prompts, greeting prompt, etc.
  - The attributes concerned by the system's action: $S = \{ s^\beta\}$. For example, a constraining question is concerned by one particular attribute and a relaxation prompt as well.

In order to enable the UM to act correctly when prompted for relaxation or confirmation, the set of possible attributes U and the set of possible values V are extended. New attributes are added with Boolean values:

$$U = \{u^{\alpha}\} \cup \{c^{\alpha}\} \cup \{r^{\alpha}\}$$
$$V^{\alpha} = \{v_i^{\alpha}\} \qquad V^{c^{\alpha}} = \{0,1\} \qquad V^{r^{\alpha}} = \{0,1\} \tag{V.4}$$

Once the variables have been defined, the topology of the network must be identified. To do so, the conditional dependencies among variables are used:

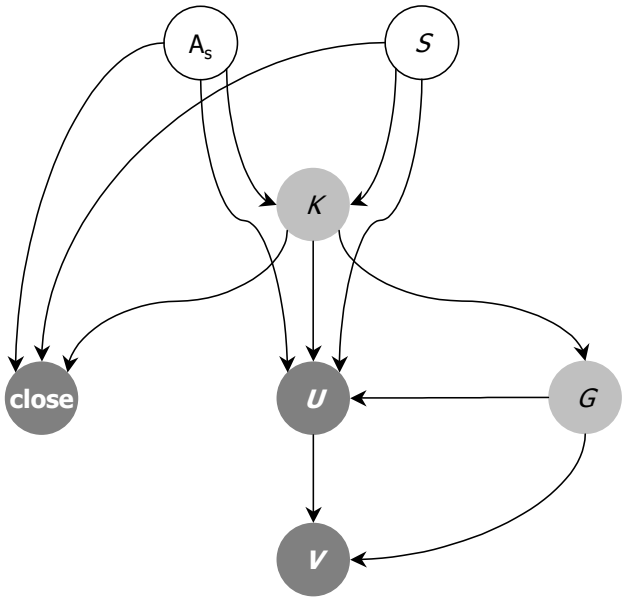


Fig. 29: A BN-based user model

- The system's action A_S and the associated attributes S are independent variables. Thus, the associated nodes have no parent.
- If assertion prompts are possible system's action, then they can update the knowledge of the user in the framework of a collaborative dialogue. Thus, there is an arc going from node A_S to node K and another arc going from S to node K.
- The goal is considered as independent of external events. This means that the user doesn't change its goal during the interaction or if it does, it is not done immediately after having heard a system's prompt but according to a change in its knowledge. Thus, there is no arc going from A_S or S to G. Yet, there is an arc going from K to G.
- The decision to close the dialogue is taken when an unsatisfactory system's behaviour is detected. It is then conditioned by the action of the system and its attributes. It is also conditioned by the knowledge of the user that keeps trace of the history of the interaction. Arcs are then drawn from nodes A_S, S and K to the node 'close'.
- The attributes U included in the user's utterance are of course conditioned by the system's action A_S and the included attributes S and by the user's goal but also by the user's knowledge since, when simulating a mixed-initiative behaviour, the already provided

attributes have a lower probability to be provided again. Arcs are then drawn from *G*, *S*, *K* and A_S to *U*.

- The value provided by the UM for a given attribute should be consistent according to the predefined goal, thus an arc is drawn from *G* to *V* in addition to the one drawn from *U*.

The resulting network is shown on Fig. 29. On this figure, dark grey circles represent the output variables, light grey circles represent internal variables and empty circles represent external variables (system's outputs).

Of course, it is possible to enumerate all the possible states of each set of variables. Yet, each state is defined by a given configuration of individual variables that can take only discrete values. Thus, it is natural to think about replacing each node of the graph shown on Fig. 29 by a factored representation. This means that to each state variable will be affected a node. For instance, let's consider the simple example of an application dedicated to fill-in a form containing only two slots (s^1 and s^2). The system can perform four types of actions: greeting, constraining questions, confirmations and closing the dialogue. To simplify, the user's goal and knowledge only include AV pairs and counts (no priority):

Goal		Know.
Att.	Val.	Count
g^1	v^1	k^1
g^2	v^2	k^2

Table 5: UM's goal for a simple form-filling application

The factored representation of the UM for this application is shown on Fig. 30. Notice that the c^i (vc^i) nodes denote the attributes (values) of the user's utterance relative to a confirmation. We can see that it can be easily generated from task structure.

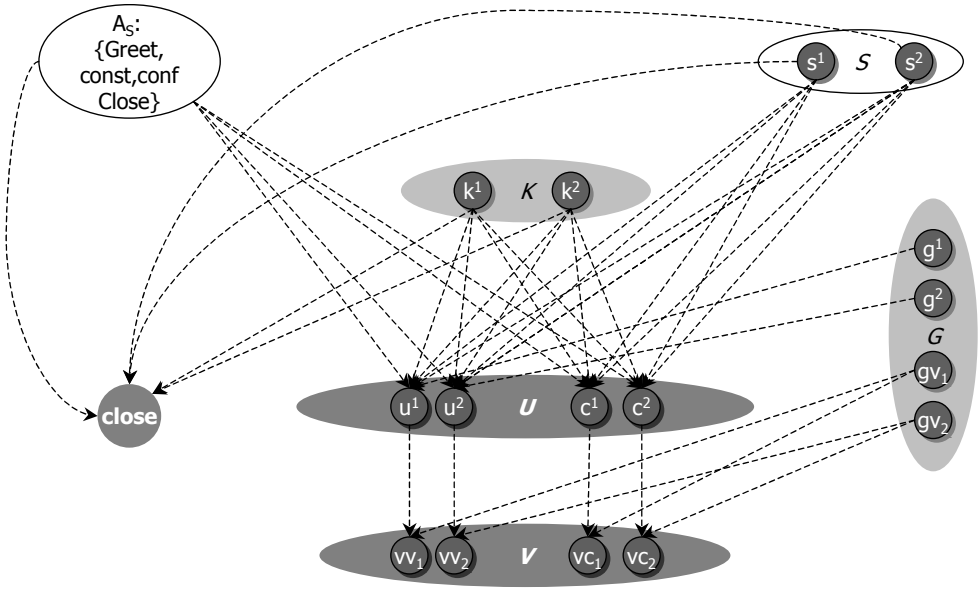


Fig. 30: Factored representation of the Bayesian UM

Since the factored BN is very connected, there are a lot of parameters to set up. Yet, an estimate of the factored representation can be automatically deduced from a small set of parameters:

- The number of possible attributes (here, the number of slots in the form)
- The number of generic actions that can be performed by the system (for example 'constraining question' is a generic action)
- The number of actions that can be affected by attributes (for example, the greeting is not affected by attributes while the constraining questions are).
- Some knowledge about the task such as the possible values for each attribute etc.

The structure can then be automatically built by following the next steps:

- Create one node for system's actions (A_S). This node can take as many values as there are possible generic actions.
- Create one node for each attribute that can affect a system's action (s^i). Each of these nodes can take only a binary value indicating whether or not the attribute affects the current action.
- Create one node for each attribute that can be a part of the UM's goal (g^i). Each of these nodes can take only a binary value indicating whether or not the attribute is present in the user's goal.
- Create a value node (gv_i) for each attribute present in the UM's goal. Each of these nodes can take as many values as there are possible values for each attribute.
- Create a priority node (p^i) for each attribute present in the UM's goal. Each of these nodes can take as many values that there are possible levels of priority (until now two levels were considered but the granularity of priorities can be increased).
- Create a counter node (k^i) for each attribute present in the UM's goal. This node can take 3 different values:
  - *low* if the attribute has never been asked by the system,
  - *medium* if the attribute has already been asked but a reasonable number of time,
  - *high* if the attribute has been asked to many times either explicitly or for confirmation.
- Create utterance attribute nodes (u^i, c^i,…) for each attribute that can be recognised by the SDS (generally the same as the attributes that can affect SDS's actions). Each node can only take binary values indicating whether or not the attribute is present in the user's utterance.
- Create utterance value nodes (vv^i, cv^i, …) for each utterance attribute node. Here, values are different for each node since they are dependent of the system's action (for confirmation there are 2 possible values while for constraining questions there are as many values than possible for the asked attribute).

- Create a ‘close’ node taking binary values and indicating whether the user might close the dialogue or not.

The connections between those nodes can also be drawn automatically:

- Draw edges from the system’s action node A_S to all the utterance attribute nodes (u^i, c^i,…) since it obviously conditions the user’s utterance: ‘1-to-many’ connection.
- Draw an edge from the system’s action node A_S to the ‘close’ node: ‘1-to-1’ connection.
- Draw edges from each system’s action attribute node (s^i) to each user’s utterance node (u^i, c^i,…): ‘many-to-many’ connection.
- Draw edges from each system’s action attribute node (s^i) to the ‘close’ node: ‘many-to-1’ connection.
- Draw edges from each knowledge node (k^i) to each user’s utterance attribute node ((u^i, c^i,…). There is an edge from k^i to u^j with $i \neq j$ to have to possibility to simulate the fact that an attribute might not be given by the user if another has not been asked before ($u^i = 0$ if $k^j =$ *low*): ‘many-to-many’ connection.
- Draw edges from each knowledge node (k^i) to the ‘close’ node: ‘many-to-1’ connection.
- Draw an edge from each goal’s attribute node (a^i) to its corresponding user’s utterance attribute node (u^i): ‘1-to-1’connection.
- Draw an edge from each user’s utterance attribute node (u^i) to its corresponding user’s utterance value node (vv_i): ‘1-to-1’connection.
- Draw an edge from each user’s goal value node (gv_i) to all the corresponding user’s utterance value nodes (vv_i, cv_i, …): ‘many-to-many’ connection. This appears to assume independency between values of the user’s utterance but not really. Actually, if there was a dependence between vv_i and vv_j (with $i \neq j$) that would be because there is a dependency between v_i and v_j. An edge should then be drawn between v_i and v_j but not between vv_i and vv_j. For example, in the train ticket booking example, we could draw an arc between the dest and the dep variables indicating that the probability P(dest = v_i, dep = v_j) = 0 for i = j. Yet, this is very task- dependent and actually reflects the task structure. It should then be derived by other means than *a priori* rules.
- Draw an edge from each priority (p^i) to each corresponding user’s utterance value nodes that suppose a modification of the goal (for example, relaxing an argument suppose to be less demanding as originally described in the goal): ‘many-to-many’ connection.

Although this structure can be enhanced like will be explained later, it is easily derived from a small set of parameters.

V.3.2. Network Parameters

There are several methods for assessing the parameters. Anyway, our purpose is first to find a way to assess them thanks to prior knowledge. A first experiment has been done for deriving those parameters from the previous goal directed model. Using this model with the computer retailing SDS, a data set has been generated and the parameters where learned from this data set.

Several conclusions could be drawn from this experiment. For instance, some of the probabilities are equals to those of the previous model:

- The probability $P(u^i = 1 \mid A_S, s^1,\ldots, s^i = 1, s^n, k^1,\ldots, k^n, g^1, \ldots, g^n)$ in the BN equals the probability $P(u^\alpha \mid s^\beta, k_t, g)$ of the previous model for $u^i = u^\alpha$ and $s^\beta = s^j$. This is obviously not surprising.
- The probability of closing the dialogue $P(\text{close} \mid A_S, s^1,\ldots s^n, k^1,\ldots, k^n)$ in the BN model is equal to $P(\text{close} \mid s^\beta, k_t, g)$ in the previous model. For example, if s^i equals 1 and k^i = *high* the probability is maximum whatever the other variables.
- The user's goal attributes and values are equally probable as well as priorities.
- The probabilities of the user's utterance values $P(vv^i \mid u^i, v_i)$ are equal to $P(V|A, g)$ in the previous model.
- The probabilities for confirmation and relaxing are also equals.

Yet some probabilities have been replaced. It is interesting to see that mixed-initiative is easier to compute within the BN framework since the probability of each attribute to be present $P(u^i = 1)$ in the user's utterance can be inferred without using $P(n \mid s^\beta, k_t, g)$ and $P(A)$.

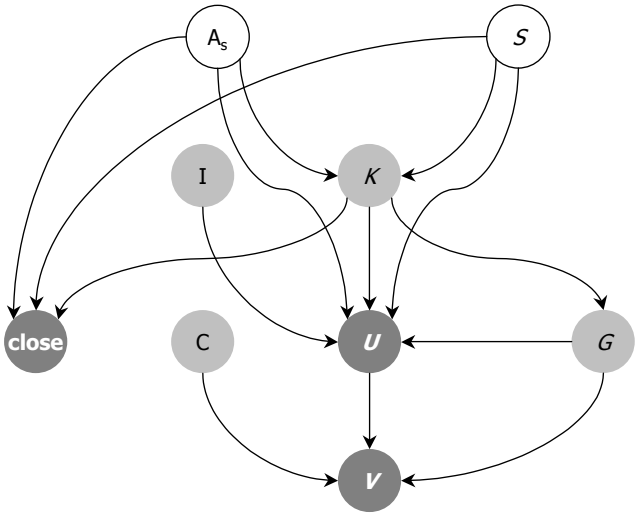


Fig. 31: Bayesian model for cooperativeness and intiative

Thus, learning parameters from the previous model is a method for assessing the parameters of the BN model but since lots of parameters are equals, it is natural to assess them directly. Yet, mixed-initiative should be simulated by assessing values for

probabilities of the form $P(u^i = a_i \mid As, s^1,\ldots, s^j = 1, s^n, k^1,\ldots, k^n, a^1, \ldots,a^n)$ where $i \neq j$. To model the initiative and the cooperativeness of the user, it is proposed to add two nodes to the network (Fig. 31). A node C will give a degree of cooperativeness while a node I will provide information about the degree of initiative (that can also be considered as a level of expertise) for the user. Each node can take a finite number of values comprised between 0 and 1 and will affect probabilities like follows:

- The probability $P(u^i = 1 \mid As, s^1,\ldots, s^i = 0, s^n, k^1,\ldots, k^n, a^1, \ldots,a^n, I)$ of providing an non-requested argument is equal to I. Thus, the higher is I the higher is the initiative (or the level of expertise) of the UM.
- The probability to provide the value contained in the goal is given by:

$$P(vv_i = v_i \mid u^1, \ldots, u^i = 1, u^n, vg_1, \ldots, vg_i = v_i, C = c) = 1 - c$$

- The probability to give another value than the one contained in the goal is giver by:

$$P(vv_i = v_i \mid u^1, \ldots, u^i = 1, u^n, vg_1, \ldots, vg_i = v_j, C = c) = \frac{c}{|V|} \quad \text{(with } i \neq j\text{)}.$$

Moreover, the level of initiative (or expertise) is used to set the level in the counts above which it is considered that the argument has been ask to many times. For example, for I = 1, the level is 1 (a very expert user only accept to be asked once for an argument) and for I = 0, the level is unlimited.

V.3.3. Enhancing the Structure

Until now, the task structure has not been used to model the user. Yet, several studies showed that a task-oriented dialogue often follows the same structure than the task itself. In our case, we supposed that each goal variable was independent from others. This is not always the case and dependencies among goal variables might modify the user's behaviour. Then, the task structure can be encoded in the user's goal structure (Fig. 32)

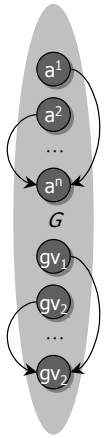


Fig. 32: Structured Goal

For instance, like already explained, in the train ticket booking example described in the beginning of section V.2, the value affected to the goal attribute variable dest exclude this value for the goal attribute variable dep. An arc should be drawn between the gv_{dest} and

the gv_{dep} variables. This kind of dependencies can be learned from some database representing the task for example. A learning algorithm has been used on the computer retailing database and it has been learned for example that the hard drive size, the ram size and the processor frequency are causes of the prise but it can be introduced by the designer that the choice of a Mac excludes the value 'Pentium III' for the processor type. Complex structure learning is nevertheless very demanding in data and is not always easy to realise. Yet, it can also be assessed by hand since it is reasonable to think that the designer of a dialogue can tell if some values are related to each other.

On another hand, edges drawn between user's goal attributes (g^i) represent implication of attributes by others or exclusion of attributes by others. That means that sometimes, an attribute needs another attribute. This can also be learned from corpus of dialogues like explain in [Meng *et al*, 1999] or handcrafted like proposed in [Wai *et al*, 2001]. These dependencies are also encoded in the arcs drawn between nodes k^i to u^j with $i \neq j$.

V.3.4. User Satisfaction

User satisfaction is very difficult to model. Yet, in the previous model, a very simple way to model user satisfaction by measuring task completion was exposed. Here it is proposed to introduce the user satisfaction in the BN framework. It is natural to think that the user satisfaction (U^S) will be dependent of the history of the dialogue session contained in *K*, of the user's level of expertise represented by I, the difference between the obtained results *R* and the initial goal variables *G* and some subjective metrics *M* about the SDS performance like the length of the dialogue or quality of the TTS system. These dependencies are shown on Fig. 33.

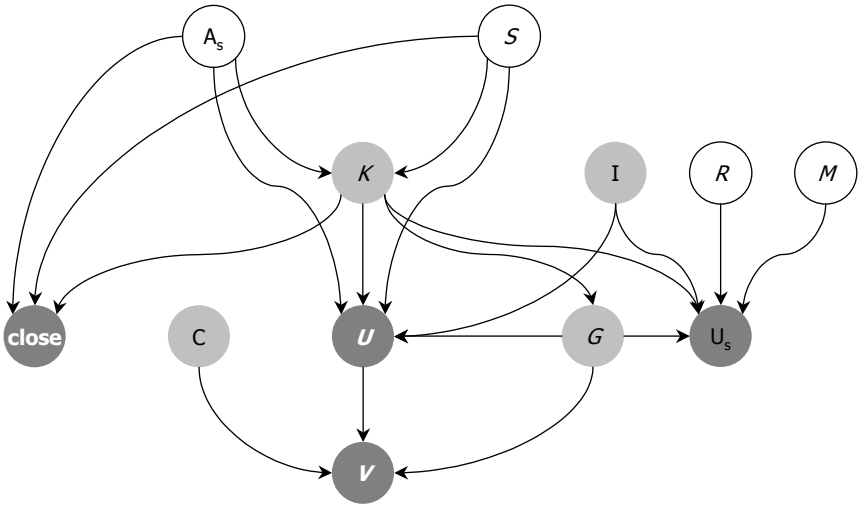


Fig. 33: Complete Bayesian UM

Of course, it is not easy to assess the parameters of this part of the network. Yet, an estimate of U^S can be obtained if it is assumed that it can only take few discrete values depending on the parents and defining some range of the user satisfaction. A real number is computed according to the following steps:

- According to the level of expertise, an *expected baseline history* is derived. This means that the higher is the level, the more the expected

history should contain variables set to *low*. The actual history is then compared to the expected one. Typically, a numerical weight is associated to each possible value of the k^i variables and the distance between the actual history and the expected one is computed. This distance is then subtracted to U^S.

- The result of the interaction is compared with the actual goal and a fixed negative number is subtracted to U^S if it is not equal (in the case of the train ticket booking, attributes are compared between the ticket and the goal, for instance). One could also use the kappa coefficient described in section II.4.5.
- According to the user's expertise level, an expected number of turns is derived (for example, lower than the number of arguments for an expert user that would prefer mixed-initiative) and this number is deduced from the actual number of turns. The result is deduced from U^S.
- Other subjective metrics can be used like the quality of the TTS system measured on a discrete scale. Those metrics has to be measured with real users.
- The satisfaction is obviously set to its minimal value if the UM closed the dialogue before its natural end.

This computation can be done for artificially generated configurations of the input values. The obtained U^S are then classified according to the possible ranges. This classification can be softened thanks to a probability distribution over the ranges (overlapping gaussian distribution for example) and the parameters relating the inputs to the output U^S can then be estimated. Notice that by using the k^i parameters, this method implicitly makes use of the number of repeated arguments, the number of confirmation. Moreover the expected history represents the ideal strategy for the user.

The ranking of the U^S value is motivated by the fact that in previous researches like [Walker *et al*, 2000] user satisfaction surveys have been used to compute a mean value for satisfaction thanks to the ranking of several subjective evaluation (like the TTS performance, the ease of task etc.). No particular weight is associated to each subjective evaluation, since it is not clear which are more important. Moreover, recent researches [Sneele & Waals, 2003] reported that adding an overall ranking statement in the satisfaction survey conducted to better results in terms of satisfaction prediction by means of objective measures obtained during the dialogue session. Thus, a single value for the overall dialogue session evaluation seems to be good enough. After all, using several rankings and average them to obtain an evaluation of the overall performance of the dialogue system makes assumptions about the importance of each ranking. It is natural to think that the user himself better realises these assumptions when asked for an overall ranking.

V.3.5. Example of Use

In the preceding paragraphs, the structure and the parameters of a Bayesian UM were described. Yet, it might not be clear how to use it for simulation purposes. Let's consider the network of Fig. 30 and state that the UM is completely cooperative (C = 0) and that it is not an expert at all (I = 0). The user's goal is depicted on Table 5. Let's simulate the answer of the user to the greeting when the dialogue starts. The following evidence is therefore inserted into the inference engine of the network:

As	k^1	k^2	a^1	a^2	gv_1	gv_2
greet	*low*	*low*	1	1	v_1	v_2

Table 6: Evidence for greeting action

From this, the inference engine produces probabilities $P(u^1=1)$, $P(u^2=1)$, $P(c^1=1)$, $P(c^2=1)$ and P(close=1) and their complements. According to those probabilities, it is firstly decided whether to close or not (the UM randomly choose a real number between 0 and 1, if it is lower than P(close = 1), the dialogue is closed). If it is not closed, the same process is done for choosing arguments present in the user's utterance. Supposing that u^1 is selected to be present in the utterance, the next evidence is then processed:

u^1	u^2	gv_1	gv_2
1	0	v_1	v_2

Table 7: Evidence for finding out values

Inference hopefully produces a null probability for all values of vv_2 but a maximum probability of 1 for the value $vv_1 = gv_1 = v_1$ since we supposed the user to be cooperative. This process is repeated all along the dialogue to simulate user's utterances. Of course, using probabilities set to 1 and 0 everywhere in the network is useless and should be avoided.

It should be noticed that this method produces user utterances according to the probabilities encoded in the network and does not use a threshold applied on the probabilities to state whether or not a given attribute should be uttered. It can then produce a variety of different utterances for the same parameters and the same evidence as far as non-null probabilities are affected to each attribute given the evidence. This models inter-variability between users belonging to a same group and modelled by a same BN.

Moreover, this model also allows for the emission of empty utterances that might also occur in real dialogues either because the user didn't understand or didn't ear the system's query or prompt.

Using this method, the desired probability $P(u_t \mid g, sys_t, s_t, n_t)$ for user modelling described in IV.17 can be computed.

V.4. Conclusion

This chapter described several parametric UMs. Each of these models has lots of parameters that can be learned from data or handcrafted by using a smaller set of parameters and some simple rules.

Since the BN framework is suitable for combining prior knowledge about task structure and user's behaviour with parameter learning capabilities, it seemed to be a very good candidate for user modelling in the framework of SDS prototype design. Moreover, the following chapters will demonstrate that the Bayesian UM can be used in other very useful ways.

A simple set of parameters can be used to generate a Bayesian UM for a given task:

- The number of possible attributes.
- The number of generic actions that can be performed by the system.
- The actions that can be affected by attributes.
- The possible values for each attribute.
- The level for the counts associated to each value of the expertise level.

Some other parameters might be provided for user satisfaction computation:

- The expected history profile for each level of expertise.
- The expected length or the dialogue for each level of expertise.

By using random values for the level of expertise and the cooperativeness, this model can generate a large amount of different dialogues. On another hand, fixing those values allows to generate dialogues for a given population of users.

This model can also supply a metric usable for dialogue evaluation since it computes a rough estimate of user satisfaction according to evaluation of the strategy.

Chapter VI:
Input and Output Processing Simulation

VI.1. Input Processing Simulation

Section II.3 described the general architecture of an SDS and the different subsystems composing the inputs processing part of the system Fig. 34, namely the ASR subsystem (translating acoustic speech signal or its representation u_t into a word sequence $w_t = \{w_{t,1},\ldots,w_{t,N}\}$) and the NLU subsystem (creating a mapping between the word sequence w_t and a sequence of concepts leading to observation o_t).

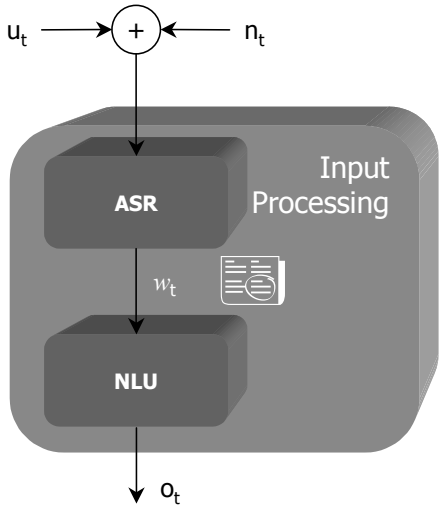


Fig. 34: Inputs processing subsystems

Since the ASR problem is a maximisation process, the 'input processing' term of expression IV.17 can be expressed as:

$$\begin{aligned} \underbrace{P(o_t \mid u_t, a_{u,t}, s_t, n_t)}_{\text{Inputs Processing}} &= \max_{w_t} P(o_t, w_t \mid u_t, a_{u,t}, s_t, n_t) \\ &= \max_{w_t} \underbrace{P(w_t \mid u_t, a_{u,t}, s_t, n_t)}_{\text{ASR}} \cdot \underbrace{P(o_t \mid w_t, u_t, a_{u,t}, s_t, n_t)}_{\text{NLU}} \end{aligned} \quad \text{(VI.1)}$$

Assuming that the NLU process does not depend on the user's utterance (but on the recognised word sequence) and the noise, last expression can be written as follow:

$$\underbrace{P(o_t \mid u_t, a_{u,t}, s_t, n_t)}_{\text{Inputs Processing}} = \max_{w_t} \underbrace{P(w_t \mid u_t, a_{u,t}, s_t, n_t)}_{\text{ASR}} \cdot \underbrace{P(o_t \mid w_t, a_{u,t}, s_t)}_{\text{NLU}} \quad \text{(VI.2)}$$

The remainder of this section describes methods to approximate the two terms of expression VI.2, though it mainly focuses on the first term. It gives a data-driven but not corpus-based approach.

VI.1.1. ASR System Model

The first term of expression VI.2 expresses the conventional speech recognition problem of translating a user utterance u_t in a sequence of words w_t belonging to the set of all possible sequences $W = \{w_i\}$. As explained in section II.3.1, the ASR process is statistical process during which errors can occur. A good model of an ASR system should thus provide a good prediction of error occurrences.

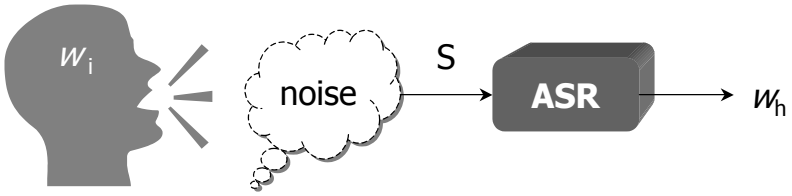


Fig. 35: High-level view of the ASR process

The ASR process can roughly be described as on Fig. 35 were the user speaks the intended word sequence w_i by producing the acoustic speech signal S. The ASR system then decodes the acoustic signal and produces the hypothesis that the word sequence w_h has been uttered. Of course there are many acoustic representations of a given word sequence because of intra- and inter-speaker variability but also because of noise. Thus, the mapping from w_i to S is one-to many. On another hand, several speech signals will result in a same decoded word sequence hypothesis and the mapping from S to w_h is then many-to-one.

Given this high-level representation of the ASR process, an ASR error occurs when w_i is different of w_h. Typical errors are insertions, deletions or substitutions in the resulting word sequence. Each error can then be considered as a confusion C_{ij} between a sequence $w_i = \{w_{i1}, \ldots, w_{in}\}$ actually uttered and a sequence $w_j = \{w_{j1}, \ldots, w_{jm}\}$ with:

- n < m if deletion occurred,
- m > n if insertion occurred,
- m = n and there is a non empty set of indices $L = \{l\}$ for which $w_{il} \neq w_{jl}$ if substitution occurred.

In equation VI.2, the term associated with the ASR process is conditioned by the noise. Including directly noise models in the estimation process is not straightforward at all. Anyway, one can easily understand that noise addition will result in additional confusions C_{ij}^{n} (confusion between sequence w_i and w_j because of noise). Thus, let's write the probability of observing the word sequence w_k at time t given the noise by:

$$P(w_t = w_k \mid u_t, a_t, s_t, n_t) = \sum_{j \neq k} P(C_{jk}^n \mid n_t) \cdot P(w_t = w_j \mid u_t, a_t, s_t) + \left(1 - \sum_{j \neq k} P(C_{jk}^n \mid n_t)\right) \cdot P(w_t = w_k \mid u_t, a_t, s_t) \quad \text{(VI.3)}$$

with $P(C_{jk}^n \mid n_t)$ being the probability of observing the word sequence w_k when the word sequence w_j would have been recognized without noise.

The term associated to pure ASR process without noise can then be expressed as:

$$P(w_t \mid u_t, a_t, s_t) = \frac{P(u_t \mid w_t) \cdot P(w_t \mid a_t, s_t)}{P(u_t \mid a_t, s_t)} \quad \text{(VI.4)}$$

And the ASR problem is to maximize this expression. Comparing this expression to equation II.4, one can see that:

- $P(u_t \mid w_t)$ is the acoustic likelihood of utterance u_t given the word sequence w_t (acoustic model).
- $P(w_t \mid a_t, s_t)$ is the Language Model (LM). It is a particular feature of SDSs to include a dependence on the action and state to the probability of occurrence of a given word sequence. Indeed, the LM can be changed at each turn in an SDS, making it highly context-dependent in order to increase ASR performance. As said in section II.3.1, the LM can be provided as an FSG, which can be handcrafted for each context, limiting valid word sequences to only a certain number at a given turn t.
- The term $P(u_t \mid a_t, s_t)$ supposes that the *a priori* user's utterance is conditioned by the context (a_t, s_t), as the system's doesn't know the interpretation of it's utterance sys_t by the user and the user's goal. All utterances are equally probable.

Even without noise, speech recognition is an error prone process. Thus, there exist confusions between word sequences C_{ij}^S (confusion between sequence w_i and w_j because of the system) due to the system itself.

Several proposals for ASR error modelling in the framework of SDS design exist in the literature but this problem is generally considered as a side-problem. For example, the dialogue simulation proposed in [Levin *et al*, 2000] doesn't take ASR errors into account. In [Hone & Baber, 1995], the ASR errors are modelled by a fixed substitution error rate. A more complex error modelling method is proposed in [Scheffler & Young, 1999]. Errors are modelled taking into account the context but error probabilities are extracted from a corpus. If LMs are modified after the data collection for example, the error model needs to be updated by a new data collection.

Making error model context-dependent is relevant since the ASR process is context-dependent according to expression (VI.4). The dependence to the context is embedded in the LM, thus it is reasonable to think that the errors occurring during the ASR process are related to the LM's properties (as it defines the set of possible word sequences). For instance, it is natural to think that the larger the allowed vocabulary is, the higher the error rate is. Then the probability of occurrence of an ASR error is defined by:

$$P(\varepsilon_t \mid u_t, a_t, s_t) \quad \text{(VI.5)}$$

However, the purpose of the simulation environment is to generate a set of dialogues having the same statistical properties than a dialogue corpus obtain by releasing the SDS to human users. Thus, only the following expression is of interest:

$$\sum_{u_t(a_t,s_t)} P(\varepsilon_t \mid u_t, a_t, s_t) \cdot P(u_t \mid a_t, s_t) = P(\varepsilon_t \mid a_t, s_t) = P_\varepsilon(LM_t) \quad \text{(VI.6)}$$

In this equation, LM_t is the language model used at time t and $P_\varepsilon(LM_t)$ is the probability of occurrence of an ASR error when using LM_t. The remainder of this section is dedicated to the estimation of the probability $P(\varepsilon_t \mid a_t,s_t)$ of occurrence of an ASR error in a given action-state configuration (thus using a given LM) and to the estimation of some metrics an ASR system can supply.

Task Classification

In a first attempt to adapt ASR error modelling to the LM, it is proposed to distinguish several recognition tasks for which specific LMs are associated. Indeed the ASR system can be used to recognize Booleans, digits, numbers, dates, isolated words, connected words, unrestricted continuous speech, etc. For each of those tasks, the recognition performances can be expressed with a set of parameters. All the parameters are estimated by collecting data covering efficiently the given task. Each recognition task will be denoted *T*. It is important to notice that recognition tasks are not related to the SDS task thus the data collection is not relying on a release of the SDS. It is obvious for Booleans, digits, numbers or date tasks, which are of course recognition tasks that can be found in any SDS task. But it is not obvious for isolated words, connected words or unrestricted continuous speech. Indeed, vocabularies and LM for those tasks can be very different but the following discussion assumes that an average value for each parameter can be obtained for each task by performing measures on several different test sets representative of the studied task. It is a strong assumption but relative performance of tasks are more important than their absolute real performance in the point of view of simulation for SDS evaluation and strategy learning.

In the framework of the pure simulation process, the first interesting parameter is the Word Error Rate (WER). The WER is very popular as a method for rating speech recognition performance and as a measure of goodness of LMs. It is a percentage and for a particular task *T* the word error rate WER_T is defined as follow:

$$WER_T = \frac{S_T + D_T + I_T}{|\text{ASR tests}|_T} \times 100\,\% \quad \text{(VI.7)}$$

In equation VI.7, S_T, D_T and I_T are namely the numbers of substitutions, deletions and insertions observed when performing $|\text{ASR tests}|_T$ recognition tests for the particular task *T*. This value is an estimate of the probability of occurrence of an ASR error for the recognition task T_t at time t.

$$P(\varepsilon_t \mid a_t, s_t) = P_\varepsilon(LM_t) = P_\varepsilon(T_t) \approx \frac{WER_{T_t}}{100} \quad \text{(VI.8)}$$

When running recognition tests for each particular task, metrics provided by the ASR systems can also be measured and probability distributions of the metrics can be estimated. For example, the Confidence Level (CL) is such a metric. The CL is a real number between 0 and 1, based on acoustic measurements and defining how sure the system is to have performed a good recognition [Williams & Renals 1997]. Its distribution is composed of two distinct curves (as shown on Fig. 36) respectively for

good and bad recognition results. As those curves cover each other, it is unavoidable to reject some well-recognized utterances as well as to accept few bad recognition results by defining a single CL threshold.

The Fig. 36 represents a CL distribution obtained from a real ASR system, for the 'isolated words' recognition task obtained according to the method described in [Mengusoglu & Ris, 2001]. These results are rather optimistic as it was obtained in optimal conditions (no noise etc.) and the curves would flatten in real running conditions. Since the simulation environment aims at comparing SDS or different implementations of a given SDS, absolute individual results are not so important as soon as relative differences between values obtained for different tasks are kept.

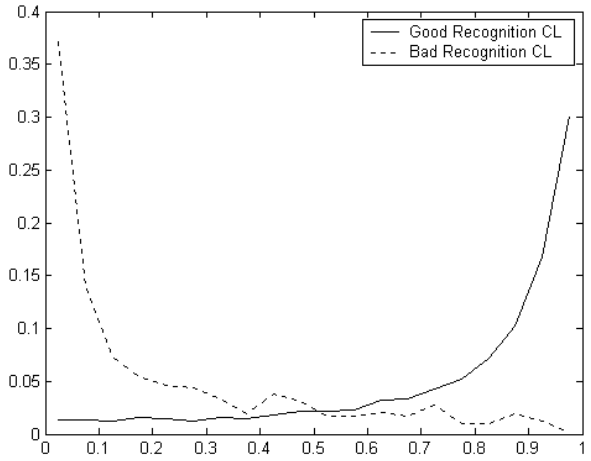


Fig. 36: Confidence Level distribution for good and bad recognitions

The input of the ASR model is the message coming from the user model. This message is a set of AV pairs standing for its intentions. In this ASR model, it will be assumed that recognition errors only affect values of the AV pairs. Indeed, only words occurring in a same context can be substituted with each other and the resulting recognised word sequence must still be correct according to the LM. Errors resulting in a complete mismatch between attributes and values or in an affectation of a correctly recognized value to an incorrect attribute will be considered as NLU errors as they actually should be detected by this sub-system.

Considering that the empty word is part of the possible substitution values (this allows for deletion errors), the simulation process for this ASR model can be summarized by the following algorithm:

```
- For each value w_i of the user's AV pair list
  {
    Choose randomly a number R_t between 0 and 1
    if R_t < P_ε(T_t)           // ASR error
        {
          1.  Substitute the current value w_i with another
              value w_k of the same nature (simulate an ASR
              error)
          2.  Produce a partial confidence level consistent
              with the "bad recognition" curve of the
              corresponding CL distribution
        }

    else                        // Good recognition
        {
          1.  Transmit the current value without
              substitution (w_k = w_i)
          2.  Produce a partial confidence level according
              to the "good recognition" curve of the
              corresponding CL distribution
        }
}
-  Transmit the new value list {w_k}
-  Generate a global confidence level for the list by
   multiplying all partial confidence levels.
```

The CL can also be seen as an observation given by the ASR model and may also be used to generate the internal state of the DM after interpretation by the task model. Thus, it is important to produce CL according to its distribution and not to always produce the mean value of the CL. Indeed, always generating the mean CL, despite what one could think, would not produce dialogues with similar statistics, as it would not result in any change in the internal DM state and some states could not be visited.

Perplexity

Although the model described in the previous section implicitly takes into account complexity features of some particular recognition tasks, it is not really formal. Moreover, its performance would more than probably decrease as the differences between the test and the real running conditions increase. In this section, we propose to evaluate more formally the complexity of the recognition task given the LM in order to estimate the WER and to generate more realistic confusion between word sequences.

One very popular measure of the complexity of the speech recognition task according to a particular LM is the (lexical) *perplexity* PP of the LM [Jelinek *et al*, 1977], which is defined by:

$$PP_{LM}^{C} = P_{LM}\left(w_1, w_2, ..., w_{|C|}\right)^{-1/|C|} \qquad \text{(VI.9)}$$

Here, PP_{LM}^{C} is the perplexity of the model LM measured on the data corpus C composed of the word sequence (w_1, $w_{2, ...,}$ $w_{|C|}$) of length |C| and $P_{LM}(.)$ is the probability of the word sequence according to the given LM (the likelihood of the model on the corpus). Perplexity can be considered to be a measure of on average how many different equally most probable words can follow any given word. The higher the likelihood of the model is, the lower is the perplexity.

It is reasonable to think that the smaller the perplexity is, the better the ASR results will be since the set of possible words at a given point is smaller. Yet, the perplexity often shows

very poor correlation with the WER [Clarkson & Robinson, 1998]. Alternative methods to predict ASR performance using metrics purely derived from the LM were proposed [Chen *et al*, 1998] but it is unclear what conclusion has to be drawn from results.

There is anyway a major obstacle to use perplexity in order to compute a corresponding WER and to use it for simulation purpose (by using the previously described algorithm, for instance). As a matter of fact, since we simulated an ASR error as a modification of values in a set of AV pairs, the problem is rather to find a way to predict the possible confusions among possible values. Here we must remember that the occurrence of a given AV pair in a particular state-action configuration results from the user model. While the occurrence of a particular attribute relies on the implementation of the user model, the occurrence of a given value only depends on the AV pairs contained in the user's goal g. As said before, all goals have equal probabilities and thus, no particular value for a given attribute is to be expected. Equal probabilities among values are then assumed. If words have equal probabilities to occur, the perplexity is only a function of the size of the vocabulary V (set of different words in the corpus) and there is no need to compute it.

Acoustic Perplexity

Perplexity has been a popular comparison measure for historical reasons because it allows LM research to develop in isolation from ASR research but it has serious shortcoming. Indeed, the possible relationship between LM perplexity and WER is due to the indication it gives about the average amount of possible words at a given point. If it increases, the added lexical entries increase the average acoustic confusability of words. But perplexity doesn't take into account acoustic similarities between candidate words at a given point, which is of course responsible for most of ASR errors. Two alternative measures to pure lexical perplexity, called *acoustic perplexity* and *synthetic acoustic word error rate* are described in [Printz & Olsen, 2000] and give significant improvement in the prediction of WER. The acoustic perplexity (APP) is defined according to the likelihood of the LM on the joint corpus (C, S) where C is a text corpus containing the written word sequence (w_1, w_2, ..., $w_{|C|}$) and S is the acoustical realisation of C s(w_1, w_2, ..., $w_{|C|}$), that is the spoken version of the written text contained in C:

$$\mathrm{APP}_{\mathrm{LM}}^{C,S} = \mathrm{P}_{\mathrm{LM}}(C \mid S)^{-1/|C|} \tag{VI.10}$$

The likelihood of the model on the joint corpus can be decomposed as:

$$\begin{aligned} \mathrm{P}_{\mathrm{LM}}(C \mid S) &= \mathrm{P}_{\mathrm{LM}}\left(w_1, w_2, ..., w_{|C|} \mid \mathrm{s}\left(w_1, w_2, ..., w_{|C|}\right)\right) \\ &= \prod_{w_h \in C} \mathrm{P}_{\mathrm{LM}}\left(w_h \mid w_1, ..., w_{h-1}, \mathrm{s}\left(w_1, w_2, ..., w_{|C|}\right)\right) \\ &= \prod_{w_h \in C} \mathrm{P}_{\mathrm{LM}}\left(w_h \mid \ell_h, \mathrm{s}(\ell_h, w_h, r_h)\right) \end{aligned} \tag{VI.11}$$

In this expression, ℓ_h is the left context of word w_h (its history) and r_h is its right context (its future). Using Bayes rule, each term of this product can be expressed as:

$$\begin{aligned} \mathrm{P}_{\mathrm{LM}}\left(w_h \mid \ell_h, \mathrm{s}(\ell_h, w_h, r_h)\right) &= \frac{\mathrm{P}\left(\mathrm{s}(\ell_h, w_h, r_h) \mid w_h, \ell_h\right) \cdot \mathrm{P}_{\mathrm{LM}}\left(w_h \mid \ell_h\right)}{\mathrm{P}\left(\mathrm{s}(\ell_h, w_h, r_h) \mid \ell_h\right)} \\ &= \frac{\mathrm{P}\left(\mathrm{s}(\ell_h, w_h, r_h) \mid w_h, \ell_h\right) \cdot \mathrm{P}_{\mathrm{LM}}\left(w_h \mid \ell_h\right)}{\sum_{w_i \in V} \mathrm{P}\left(\mathrm{s}(\ell_h, w_h, r_h) \mid w_i, \ell_h\right) \cdot \mathrm{P}_{\mathrm{LM}}\left(w_i \mid \ell_h\right)} \end{aligned} \tag{VI.12}$$

Here, $w_i \in V$ are words composing the vocabulary of all distinct words in the corpus. One interesting conclusion can be drawn by considering that all words sound the same:

$$P(s(\ell_h, w_h, r_h) \mid w_i, \ell_h) = \alpha \cdot \frac{1}{|V|} \tag{VI.13}$$

then:

$$P_{LM}(w_h \mid \ell_h, s(\ell_h, w_h, r_h)) = \frac{(\alpha \cdot 1/|V|) \cdot P_{LM}(w_h \mid \ell_h)}{(\alpha \cdot 1/|V|) \cdot \sum_{w_i \in V} P_{LM}(w_i \mid \ell_h)} = P_{LM}(w_h \mid \ell_h) \tag{VI.14}$$

and finally,

$$APP_{LM}^{C,S} = \prod_{w_h \in C} P_{LM}(w_h \mid \ell_h, s(\ell_h, w_h, r_h))^{-1/|C|} = \prod_{w_h \in C} P_{LM}(w_h \mid \ell_h)^{-1/|C|} = PP_{LM}^{C} \tag{VI.15}$$

This means that when all words sound exactly the same, the acoustical perplexity equals the lexical perplexity. In other words, this demonstrates formally that the lexical perplexity doesn't take into account acoustical confusability and that it is a big drawback since APP correlates better with WER.

WER Prediction

The word acoustic confusability is embedded in the term $P(s(\ell_h, w_h, r_h) \mid w_i, \ell_h)$ of expression VI.12. This term can be interpreted as the probability of decoding the word w_h while the intended word w_i has been uttered. As it relies on the infinite set of acoustical realisations of any word w_h in any right and left context, this term should then be approximate in order to compute APP. Yet, there isn't any existing joint corpus as large as to estimate directly these terms by empirical methods. Considering that the ASR system studied in this framework is based on a ANN/HMM model like described in section II.3.1, and that this model has been trained on a sufficiently large joint corpus (*C*,*S*), several methods for estimating $P(s(\ell_h, w_h, r_h) \mid w_i, \ell_h)$, can be investigated..

Before going further in the investigation of possible solutions to the problem of acoustic confusability let's define some assumptions. First, as we are interested in confusion occurring at a particular grammar state (values transmitted by the user), from no on in this section we will consider that the decoding of a word at a given grammar state is done independently of the surrounding spoken words. This can also be seen as the decoding of words surrounded by silence. Moreover, it will also be assumed that the spoken version of a word is independent of the previously uttered words:

$$P(s(\ell_h, w_h, r_h) \mid w_i, \ell_h) \approx P(s(w_h) \mid w_i) \tag{VI.16}$$

The right-most term of expression IV.16 is referred to as the *acoustic encoding probability* in [Printz & Olsen, 2000]. The problem is then to approximate the acoustic encoding probabilities for all pair of words. Furthermore, in order to generalize our framework, we will consider that each word of the vocabulary *V* can have multiple pronunciations, that is multiple phonetic transcriptions. The following notation will be adopted:

$$\Phi(w) = \left\{ \varphi^1(w), ..., \varphi^{|\Phi(w)|}(w) \right\} \tag{VI.17}$$

where $\Phi(w)$ is the set of all possible phonetic transcriptions (or lexeme) for word w and $\varphi^{\alpha}(w)$ is the α^{th} possible phonetic transcription (or lexeme) for word w. Each $\varphi^{\alpha}(w)$ is therefore a sequence of phonemes:

$$\varphi^{\alpha}(w) = \left\{ \varphi_1^{\alpha}(w), ..., \varphi_{|\varphi^{\alpha}(w)|}^{\alpha}(w) \right\} \tag{VI.18}$$

According to those notations, the acoustic encoding probability can be rewritten as:

$$P(s(w_h) | w_i) = \sum_{\beta=1}^{|\Phi(w_i)|} P(s(w_h) | \varphi^{\beta}(w_i)) \cdot P(\varphi^{\beta}(w_i) | w_i) \tag{VI.19}$$

with,

$$P(s(w_h) | \varphi^{\beta}(w_i)) = \sum_{\alpha=1}^{|\Phi(w_h)|} P(s(\varphi^{\alpha}(w_h)) | \varphi^{\beta}(w_i)) \tag{VI.20}$$

thus,

$$P(s(w_h) | w_i) = \sum_{\alpha=1}^{|\Phi(w_h)|} \sum_{\beta=1}^{|\Phi(w_i)|} P(s(\varphi^{\alpha}(w_h)) | \varphi^{\beta}(w_i)) \cdot P(\varphi^{\beta}(w_i) | w_i) \tag{VI.21}$$

When introducing this last expression in VI.12, and thanks to the acoustic perplexity definition we have:

$$\hat{APP}_{LM} \approx \left(\prod_{w_h \in C} \frac{\sum_{\alpha=1}^{|\Phi(w_h)|} \sum_{\beta=1}^{|\Phi(w_h)|} P(s(\varphi^{\alpha}(w_h)) | \varphi^{\beta}(w_h)) \cdot P(\varphi^{\beta}(w_h) | w_h) \cdot P_{LM}(w_h | h_h)}{\sum_{w_i \in V} \sum_{\alpha=1}^{|\Phi(w_h)|} \sum_{\beta=1}^{|\Phi(w_i)|} P(s(\varphi^{\alpha}(w_h)) | \varphi^{\beta}(w_i)) \cdot P(\varphi^{\beta}(w_i) | w_i) \cdot P_{LM}(w_i | h_h)} \right)^{\frac{1}{|C|}} \tag{VI.22}$$

Starting from this approximate expression of the acoustical perplexity we can draw some conclusions. On one hand, let's have a look to the LM contribution. Indeed, if one wanted to simplify the expression, one could write:

$$P_{LM}(\varphi^{\beta}(w_h) | h_h) = \sum_{w_j \in V} P(\varphi^{\beta}(w_h) | w_j) \cdot P_{LM}(w_j | h_h) \tag{VI.23}$$

If we consider that $\varphi^{\beta}(w_h) \notin \Phi(w_j) \ \forall \ j \rightarrow P_{LM}(w_j \mid h_h) \neq 0$, that is that there is no homophonous word in the set of the possible words at a given grammar state, we have:

$$P_{LM}(\varphi^{\beta}(w_h) | h_h) = P(\varphi^{\beta}(w_h) | w_h) \cdot P_{LM}(w_h | h_h) \tag{VI.24}$$

and equation VI.22 becomes

$$\hat{APP}_{LM} \approx \left(\prod_{w_h \in C} \frac{\sum_{\alpha=1}^{|\Phi(w_h)|} \sum_{\beta=1}^{|\Phi(w_h)|} P(s(\varphi^{\alpha}(w_h)) | \varphi^{\beta}(w_h)) \cdot P_{LM}(\varphi^{\beta}(w_h) | h_h)}{\sum_{w_i \in V} \sum_{\alpha=1}^{|\Phi(w_h)|} \sum_{\beta=1}^{|\Phi(w_i)|} P(s(\varphi^{\alpha}(w_h)) | \varphi^{\beta}(w_i)) \cdot P_{LM}(\varphi^{\beta}(w_i) | h_h)} \right)^{\frac{1}{|C|}} \tag{VI.25}$$

The log of expression VI.25 is generally easier to compute.

On another hand, the term $P(s(\varphi^{\alpha}(w_h)) | \varphi^{\beta}(w_i))$ should be regarded as the probability that the acoustic realisation of the intended phoneme sequence $\varphi^{\beta}(w_i)$ will be captured by the

acoustic model M_h of the phoneme sequence $\varphi^{\alpha}(w_h)$ and decoded by the ASR system as being an acoustic realisation of the word w_h while it should be captured by the acoustic model M_i of the phoneme sequence $\varphi^{\beta}(w_i)$ and decoded as being an acoustic realisation of the word w_i.

Remembering that the ASR system is based on the composition of sub-word unit generative models (HMMs), the term $P(s(\varphi^{\alpha}(w_h))|\ \varphi^{\beta}(w_i))$ could also be regarded as the probability that a model M_i encoding the sequence $\varphi^{\beta}(w_i)$ will generate an observation sequence $s(\varphi^{\beta}(w_h))$ for which the likelihood on the model M_h representing the sequence $\varphi^{\alpha}(w_h)$ is maximum. With this interpretation, a method to approximate the probability $P(s(\varphi^{\alpha}(w_h))|\ \varphi^{\beta}(w_i))$ would be to generate a large number of sequences from all the models, measure the likelihood of each sequence given each model and derive a metric enabling to express the similarity of two models. This method family will be referred as *model sampling* methods. Some basics of these methods can be found in [D'Orta *et al*, 1987].

The problem of estimating $P(s(\varphi^{\alpha}(w_h))|\ \varphi^{\beta}(w_i))$ could also be solved analytically (*analytical* methods) if an objective metric between models M_i and M_h could be defined when knowing their parameters (see section II.3.1). Research reported in [Lyngsø *et al*, 1999], for instance, defines the co-emission probability of two models in the framework of genetic sequence alignment and derives a similarity measure between HMMs. Other recent researches have been made in this sense in other application fields of HMMs [Bahlmann & Burkhardt, 2001], showing the value of such analytical development.

Nevertheless, both sampling and analytical methods make use of the acoustical model of the studied ASR system and thus share the same assumptions about the acoustic data. Moreover, it is based on the internal functioning of the actual ASR system and particularly on the model parameters, which are not always known (particularly if a commercial product is used). It is also highly cumbersome.

Another method family can be developed if it is assumed that model similarities arise because of similarities in the data they are supposed to model. In this particular case, the models are used for recognition of phoneme sequences from acoustic data. Thus an estimate of $P(s(\varphi^{\alpha}(w_h))|\ \varphi^{\beta}(w_i))$ should be derivable from a metric defined on acoustic properties of the phoneme sequences. In this framework, the metric could define a distance or a similarity between phoneme sequences represented by a phonetic transcription. From now on, we will assume that the phonetic transcription(s) of each word of the vocabulary V is known. In general, phonetic transcriptions can be obtained from orthographic transcriptions by grapheme-to-phoneme techniques mainly developed for TTS synthesis (see section II.3.4).

The problem is then to find a metric between two sequences of symbols φ_i coming from an alphabet A, which can be seen as a sequence alignment problem. One of the most popular methods for measuring the difficulty of aligning two sequences is the *edit distance* that is the minimum number of edit operations (namely substitutions, insertions and deletions of symbols) required to transform one sequence into the other [Levenshtein, 1966]. The edit distance can be efficiently computed by a Dynamic Programming (DP) algorithm. In this framework, to each edit operation is associated a cost, for deletions (insertions) there exists a single mapping between the symbol and the cost c_i^d (c_i^i) of deleting (inserting) the symbol φ_i while for substitutions, a $|V|x|V|$ substitution cost matrix (SCM) has to be built:

$$SCM = \begin{bmatrix} c^s_{11} & \cdots & c^s_{1j} & \cdots & c^s_{1|V|} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ c^s_{i1} & \cdots & c^s_{ij} & \cdots & c^s_{1j} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ c^s_{|V|1} & \cdots & c^s_{|V|j} & \cdots & c^s_{|V||V|} \end{bmatrix} \quad \text{(VI.26)}$$

where c^s_{ij} is the cost for substituting symbol φ_i with symbol φ_j. The role of the DP algorithm is to optimise the alignment process by minimizing the overall cost. Thus, our primary problem splits into two sub-problems:

- What are the better costs c^d_i, c^i_i and c^s_{ij} to associate with each edit operation knowing that symbols are phonemes of a given language?
- When the edit distance is computed, how to transform it in order to use it in the acoustic perplexity estimate computation?

To answer to the first question, several approaches can be considered given that acoustic data are available or not. Yet, the standard method is to adopt an arbitrary constant cost for each kind of edit operation. The substitution cost is generally considered to be higher than deletion and insertion costs but it can reasonably thought that such an arbitrary cost policy would provide poor results. Indeed it doesn't take into account particular features of the studied symbols, namely the phonemes. Such a cost matrix can be represented like on Fig. 37. Phonemes are numbered from 1 to 35 according to Appendix A: Table 37 (vowels and semivowels are numbered from 1 to 18 and consonants are numbered from 19 to 35), a null phoneme is added in position 36 to the set of usual phonemes in order to include insertion and deletion costs in the computed matrix.

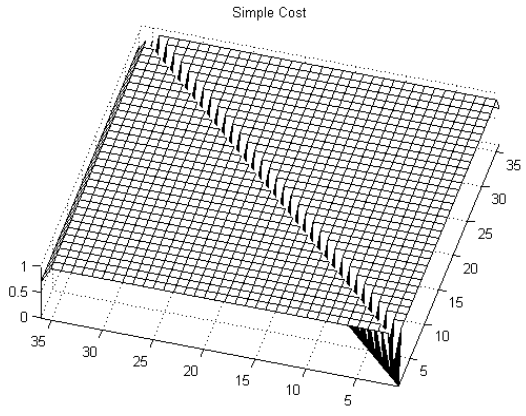


Fig. 37: Simple cost matrix

On another hand, we could take the term 'distance' in its straight sense and use methods similar to those used in previous ASR systems. This means using prototypes of acoustic realisation of each phoneme (or more realistically, prototype acoustic feature sets representing those acoustic realisations). Considering that for each phoneme φ_i we have a prototype feature set f_i of length $| f_i |$, we could use those data to compute the cost of each edit operation. A simple cost for substituting φ_i to φ_j would be $|| f_i | - | f_j ||$ (absolute value of the length difference) which would make sense since substituting long phonemes with shorter ones is less likely. Another more realistic cost would be the distance between the feature vector sets computed thanks to a DTW algorithm. This would not really result in

the same distance than if computed between the two large feature vector sets obtained by concatenating sets corresponding to the phonemes since we consider insertions and deletions (for which the target phoneme is the empty phoneme).

The cost of each edit operation can also be estimated from available data by probabilistic methods. One first method would be to associate the log-likelihood of each edit operation to the corresponding cost. Notice that what should be measured is the probability of a particular symbol substitution, insertion or deletion to occur and not the probability of a word to be mismatched with another. The maximum likelihood estimate of this probability can be computed by measuring the frequency of each edit operation. On another hand, the edit distance could be directly learned from data like exposed in [Ristad & Yianilos, 1998]. On Fig. 38 are shown the results obtained with this method on the BDSons database [Carre *et al*, 1984]. The database contained a vocabulary of 538 different isolated words spoken by male and female users in more than 8000 one-word recorded utterances. The ASR system used was trained on the BREF corpus [Lamel *et al*, 1991] but surprisingly gave very poor results since the WER was around 45% on the complete database. Yet some conclusions can be drawn from this test. For example, we can see that vowels are more often confused with each other and not with consonants (and *vice-versa*), deletions and insertions are not equally probable for all phonemes and it is also important to notice that the matrix is not symmetric.

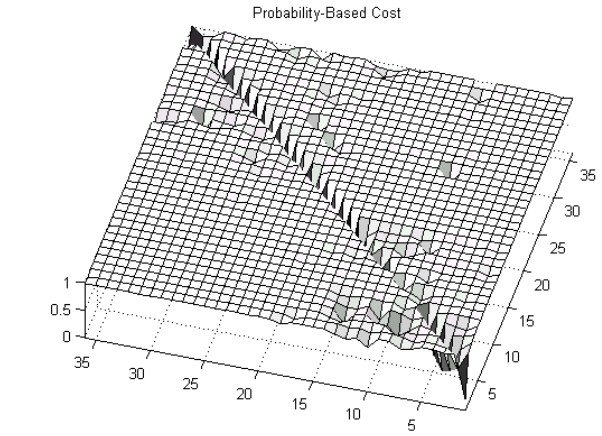


Fig. 38: Probability-based cost

Nevertheless, it is difficult to take further conclusions from this experiment since a lot of very different words were confused, lots of phonemes were deleted instead of substituted and there were also a lot of homophonous words in the corpus. Moreover, lots of substitutions that occur in general tasks had a null probability to occur according to this experiments.

Anyway, a practical cost should only be based on phonetic transcriptions. Yet, all phoneme substitutions should not be assigned the same cost as well as all deletions or insertions since phonemes do not all have the same acoustical properties. Particular features of each phoneme resulting in similar observable modifications in the acoustic signal can be derived from articulatory features (see Appendix A: Table 37 and Table 38). Thus, a better substitution cost estimate could be derived from the number of different acoustic features between the two substituted phonemes.

Once again, every articulatory feature differences should not be considered as having equal costs because their acoustic realisations are not equally distant. For example, confusing a vowel and a consonant should deserve a higher cost than a substitution

between two vowels or two consonants differing by another feature. In the following, we will mainly discuss the case of French phonemes although extension to other languages should not be too complex.

According to linguistic conventions, which are confirmed by the test on the BDsons database (Fig. 38), we can state the following rules (phonemes transcriptions are written according to the SAMPA (Speech Assessment Methods Phonetic Alphabet) phonetic alphabet):

- Confusions between vowels and consonants have a higher cost
- Confusions between consonants that only differ by the articulatory point (the location in the mouth where the sound is produced) are more likely and even more likely if the articulatory points are close to each other.

  Ex: artère [a R t e r] → [a l t e r] (altère)

- Confusions between vowels that only differ by the aperture degree are more likely (see Table 38) and confusion occur more often by augmenting the aperture (thus, the confusion matrix will be asymmetric).

  ex: aimer [e m E] → [EmE]

- Deletion or insertion of liquid consonants are more likely than other consonants.

  ex: altère [a l t e r] → [a t e r] (à terre)

- Deletions or insertions of the phoneme @ (shwa) are more likely than other vowels.

Although it is more difficult to implement, we argue that contextual information should also be taken into account in order to associate a cost to edit operations. Indeed, the following rules can also be asserted:

- Deletions are more likely at the beginning and the end of a word and should then deserve a smaller cost.
- Insertions or deletions of consonants are more likely immediately before or after another consonant.

  ex: à terre [a t e r] → [a l t e r] (altère)

- A vowel can be articulated with a different aperture degree when it is close to another vowel differing only by this feature ('close to' means in the surrounding syllables and not further). It is even more likely when a tonic accent is on the other vowel.

  ex: aimer [e m E] → [E m E]

- A voiced consonant can become an unvoiced consonant when it is just before or after to a unvoiced consonant

  ex: obturer [o b t y r E] → [o p t y r E]

These rules are of course not exhaustive but they provide good results, as we will see. Moreover, this takes into account acoustic proximity not only for ASR models but also probable bad pronunciations that can result in bad recognitions (in case the modified pronunciation is not known by the ASR system).

Given that a distance based on the articulatory features is chosen, several distance definitions can be considered. We will derive our distance from the well-known 'Ratio Model' matching function [Tversky, 1977] that defined the similarity between to sets of features. If we denote the set of articulatory features of phoneme φ_i by f_i, the similarity between φ_i and φ_j phonemes can be expressed by:

$$s(\varphi_i, \varphi_j) = \frac{F(f_i \cap f_j)}{F(f_i \cap f_j) + F(f_i \setminus f_j) + F(f_j \setminus f_i)} \quad \text{(VI.27)}$$

In this expression, $F(\cdot)$ is a function applied on a set of articulatory features that assigns a weight to each feature. The weights can be assigned according to the heuristics described in the above. We can modify the 'Ratio Model' matching function in order to obtain a distance between two sets of features:

$$d(\varphi_i, \varphi_j) = \frac{F(f_i \setminus f_j)}{F(f_i \cap f_j) + F(f_i \setminus f_j) + F(f_j \setminus f_i)} \quad \text{(VI.28)}$$

According to this definition, we can assign a cost to substitution operation (proportional to the distance). This definition of the distance between two phonemes allows for asymmetric SCM, which is a desired effect since we described asymmetric costs (for aperture degrees for, instance). On Fig. 39, the difference between a simple feature-based phoneme distance and a weighted version of this distance is shown. We can see that the weighted version is asymmetric and the asymmetry arise at the same places than on the surface depicted on Fig. 38. Moreover, whatever the distance version, consonants are closer to each other than to vowels (and *vice-versa*).

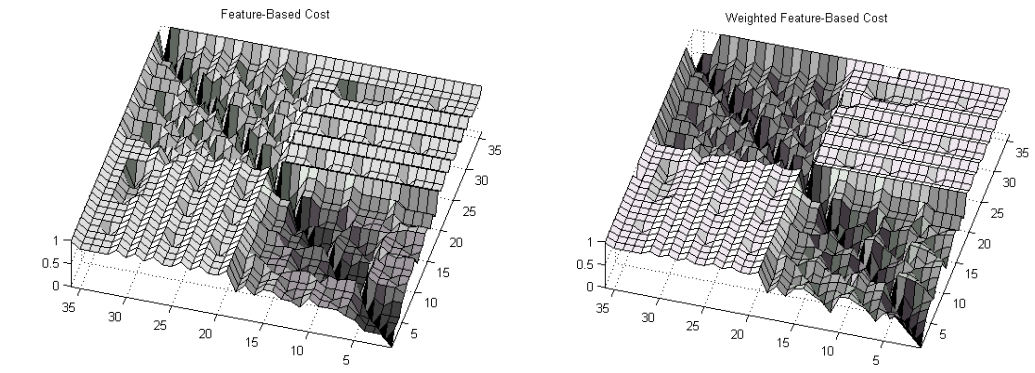


Fig. 39: Feature-based phoneme distance

Insertion and deletion costs are also adjusted according to heuristics described in the above (it is also shown on the weighted feature-based cost matrix). A DP algorithm can then be applied to phoneme sequences (word phonetic transcriptions) and an inter-word distance can be computed as the cost of the alignment between two sequences. The weighted feature-based cost matrix has been used for aligning hypothesis phoneme sequences on reference sequences resulting from the experiment on the BDsons database. The alignment was accurate in 94% of the cases (that is, handcrafted alignment was different from automatic alignment in 6% of bad recognition results).

Once the inter-word distance has been obtained, a confusion probability between phoneme sequences can be derived by simply assessing that:

$$\lambda \cdot d\left(\varphi^{\alpha}(w_h), \varphi^{\beta}(w_i)\right) \propto -log\, P\left(s\left(\varphi^{\alpha}(w_h)\right) \mid \varphi^{\beta}(w_i)\right) \quad \text{(VI.29)}$$

were λ is a constant positive parameter. Consequently, we have:

$$P\left(s\left(\varphi^{\alpha}\left(w_h\right)\right)\middle|\,\varphi^{\beta}\left(w_i\right)\right) \propto e^{-\lambda \cdot d\left(\varphi^{\alpha}\left(w_h\right),\varphi^{\beta}\left(w_i\right)\right)} \quad \text{(VI.30)}$$

This probability is used to compute the acoustic perplexity in order to estimate the WER. Results shown on Fig. 40 show a quite good agreement between estimates and real values. Yet the experiment was done on the same database than previously (BDsons) and results are quite bad (large WER).

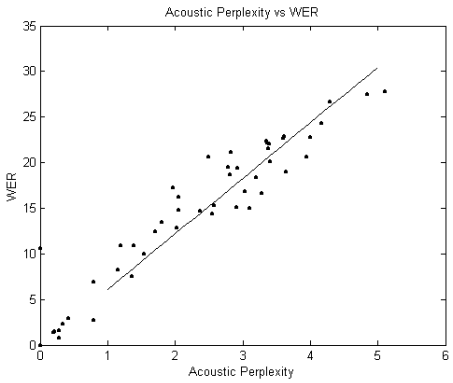


Fig. 40: Acoustic Perplexity vs. WER

Predicting the Confidence Level

In general, CL curves like those depicted on Fig. 36 can be approximate by a sum of exponential distributions:

$$P\left(CL\,|\,\lambda_i\right) \propto \sum_i \lambda_i \cdot e^{-\lambda_i \cdot CL} \quad \text{(VI.31)}$$

In order to approximate confidence level distribution for both good and bad recognition results, we have to approximate P(badCL) and P(1-goodCL) where badCL is the CL for bad recognition results and goodCL is the CL for good recognition results:

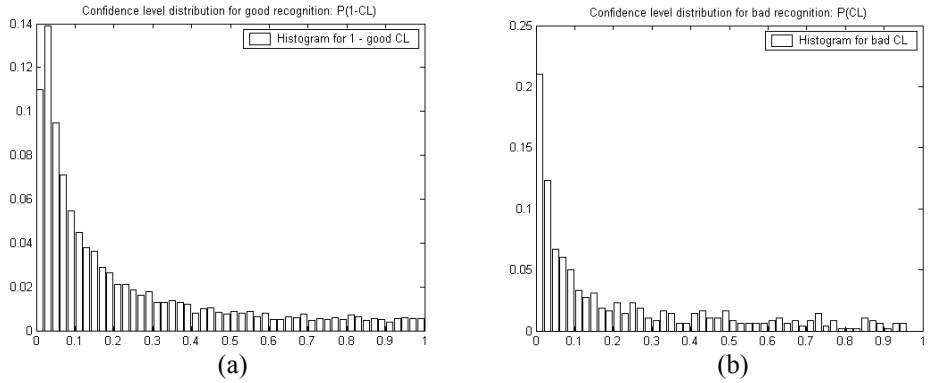


Fig. 41: Histogram distribution of (a) 1-goodCL and (b) badCL

Yet, it seems quite difficult to find the best sets of λ_i parameters with a set of phonetic transcription as only clue. Our first thought was to simplify the problem by approximating the curves by a single exponential curve:

$$P\left(CL\,|\,\lambda\right) \propto \lambda \cdot e^{-\lambda \cdot CL} \quad \text{(VI.32)}$$

A starting estimate for parameter λ could be computed on a real data set $\{CL_1,\ldots,CL_M\}$ by ML estimation:

$$\hat{\lambda} = \frac{\sum_{i=1}^{M} CL_i}{M} \tag{VI.33}$$

This estimate should serve as a starting point to compute new curves thanks to some function of inter-word distance inside a given vocabulary. Results of parameter estimation on real data are shown on Fig. 42. This figure shows that high confidence level probabilities are widely underestimated for both curves by the single exponential approximation.

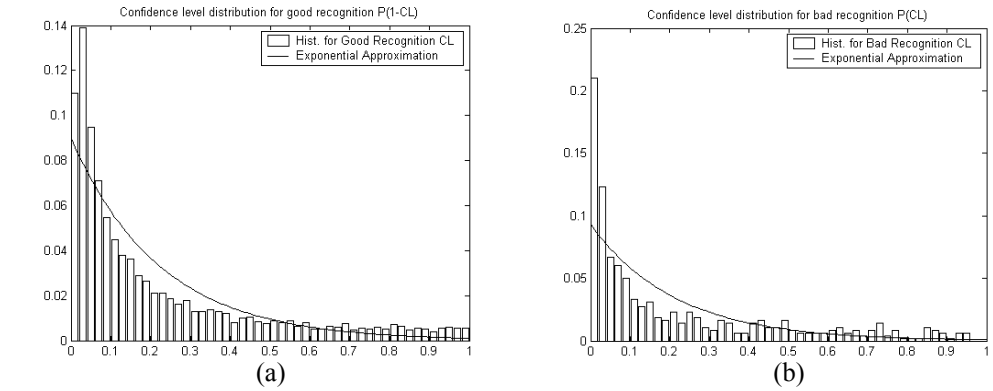


Fig. 42: Exponential approximation (a) for bad recogntion, (b) for good recognition

We then choose to model these distributions by the sum of two exponential distributions: one computed on the lower half of the data and the other on the upper half:

$$P(CL \mid \lambda_{low}, \lambda_{high}) = \lambda_{low} \cdot e^{\lambda_{low} \cdot CL} + \lambda_{high} \cdot e^{\lambda_{high} \cdot CL} \tag{VI.34}$$

This assumption doubles the log-likelihood of the data upon the obtained distribution and the curves fit better the data histograms:

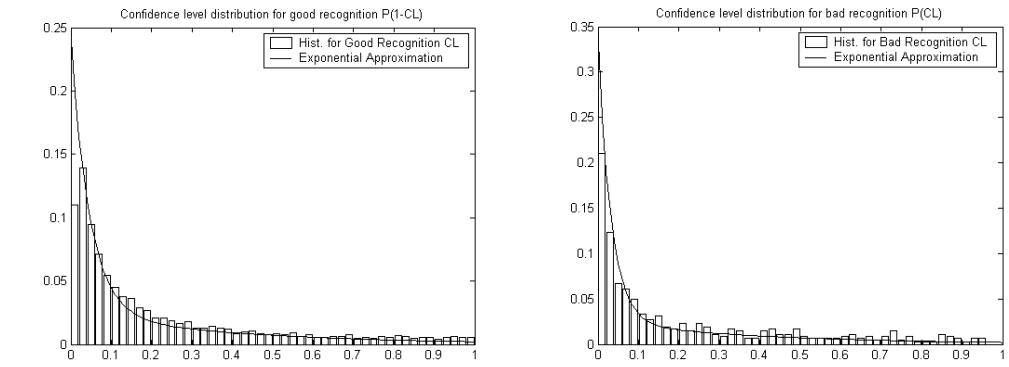


Fig. 43: Approximation by a sum of two exponential distributions

When the mean distance between the words composing the vocabulary decreases, the curves flatten and both λ_{low} and λ_{high} decrease and especially for bad recognition results. It

is not surprising since the CL measure used here is based on the priors of acoustic frames and when the distance between two confused word decreases, it probably means that phonemes with similar acoustic features has been substituted to each other. Thus the CL is less reliable. A similar effect is measured on the goodCL distribution but at a smaller level. It might be because smaller mean distance means that words in the vocabulary induce possible problems of pronunciations (since the distance can also takes this problem into account). A problem of pronunciation doesn't always result in a bad recognition result but the acoustic priors might be modified.

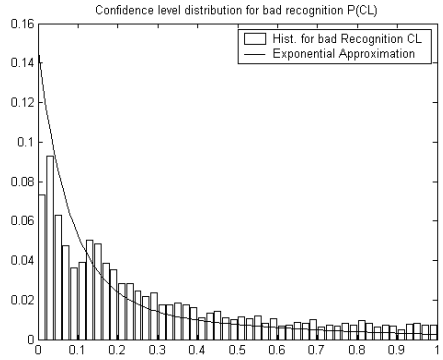


Fig. 44: Distribution of badCL with smaller mean inter-word distance

Since the modifications of the CL distribution for well-recognized words is not really significant and not well explained, it will be supposed constant. Nevertheless, the distribution for bad recognition results should be modified. It can be modified for the whole vocabulary given the mean inter-word distance or computed for each word like explain in the next sub-section.

Other Possible ASR Metrics

The computation of acoustic perplexity is a quite complex computation, especially for large vocabulary sizes. Moreover, one could need to create more realistic substitutions, not according to a global WER but according to an error rate give the word. Thus, we can think about other metrics or other methods for computing WER. Since an inter-word distance has been defined, we can apply a clustering algorithm to the vocabulary and group the words that are closer to each other. An example is shown in Table 8 and the error rate on a given word seems to correlate with the number of word in the same cluster. Once again the results obtained on the BDsons database are very bad, but since the purpose is to compare dialogues with each other, approximating the error rate on a given word by the following expression can be satisfactory:

$$\text{error rate}(w) = \alpha \, |\text{cluster}(w)| \tag{VI.35}$$

A global WER can also be computed:

$$\text{WER} = \alpha \cdot \frac{\sum_{w \in \text{V}} |\text{cluster}(w)|}{|\text{V}|} \tag{VI.36}$$

This last equation means that the mean size of a cluster in a given vocabulary is an indication of the WER.

Word	Cluster Size	WER	Words sharing the same cluster
Barre	23	95%	Bal, Balle, Dard, Gare, Par, Berre, Car, Jars, Tard, Parle, Dalle, Gale, Pal, Beurre, Bord, Bore, Gère, Guerre, Père, Char, Phare, Sar
Feinte	16	75%	Faite, Sainte, Fente, Teinte, Peinte, Quinte, Tinte, Pinte, Geinte, Fonte, Fête, Sente, Vente, Chante, Faute
Express	1	0%	

Table 8: Clusters in the BDSons Vocabulary

In the same way, the confidence level distribution can be estimated from the size of the clusters. Indeed, the confidence level distributions for a given word have the same shape as shown before. When the recognized word (and not the actually spoken word) is in a large cluster, the distributions are flattened and they are very sharp otherwise.

VI.1.2. NLU System Model

The NLU process aims at transforming a sequence of words in a sequence of concepts. According to the AVM description of the task, the NLU process affects values to attributes given the ASR results (Fig. 45). Thus, a NLU error would mainly result in associating a value to an incorrect attribute.

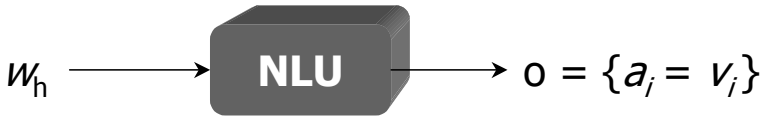


Fig. 45: NLU process

According to equation VI.2, the simulation of the NLU process involves the estimation $P(o \mid w_h, a_u, s)$. In the field of ASR, there are some standard implementations (like the hybrid ANN/HMM approach)) and new approaches are becoming rare (of course, improvements of currently used techniques are always the subject of researches). This allowed taking the general ideas of those implementations as a basis for simulation in the preceding section. Nowadays, the same conclusions cannot really be drawn in the field of NLU. Lots of different implementations can be found in the literature. Grammar-based techniques are still used in some systems [Allen, 1987]. On another hand, statistical approaches similar to those used in ASR decoding (HMM-based approaches) have also been developed and used in working systems [Pieraccini & Levin, 1992]. Recently, new statistical models for NLU have been proposed based on belief networks and showed pretty good results in several tasks [Meng *et al*, 1999]. Thus, no commonly admitted solution exists and it is difficult to model the NLU process thanks to a general implementation framework. It is thereafter proposed to make use of the model that generated the utterances as a classifier of the concepts embedded in the observed utterances.

Bayesian NLU Model

In section V.3, a Bayesian User Model was described. It is proposed to use this model to simulate NLU process. To do so, the retrieved values coming from the ASR model will be used as pieces of evidence for the inference engine of the UM.

Referring to the UM described on Fig. 30, and considering that the ASR process produced a word that can represent the value v_j when the system prompted the user with the greeting prompt. The following evidences can be introduced into the inference engine of the Bayesian UM:

As	s_1	s_2		vv_1		vv_2
greet	0	0	with	v_j	or	v_j

Table 9: Evidence for simulating the NLU process

Unless the value v_j is not a possible value for one of the tested arguments, this two different evidences will provided values for $P(u^1 \mid As = greet, vv_1 = v_j)$ and $P(u^2|As = greet, vv_2 = v_j)$. The NLU simulation sub-system will then affect the value to the attribute with the highest probability to occur in the sentence given the evidence. Nevertheless, since non-null probabilities are possible for other attributes, this might introduce errors.

The case described just before is very simple but more complex situations can occur. For example, if we consider that the system keeps trace of the dialogue history (from its point of view). This is generally done when updating its state. Considering also that another value v_k has been retrieved when the system asked to confirm the first attribute after the greeting. Then 3 more complex evidences can be introduced in the inference engine:

As	s_1	s_2	k_1	vv_1		vv_2	cv_1	cv_2
conf	1	0	*medium*	v_j	with	v_k	-	-
conf	1	0	*medium*	v_j		-	v_k	-
conf	1	0	*medium*	v_j		-	-	v_k

Table 10: Evidences taking history into account

This will provide probabilities for the 3 remaining attributes of having the value v_k. But what is important to keep in mind is that the history introduced in the evidence is the history kept by the system, which is not always the same as the user's one. This might also be a source of errors.

Finally, when several possible values have been retrieved by the ASR system, each value is used to build evidence, which is processed like said just before. The attribute providing the highest probability is kept for each piece of evidence separately.

This method can also provide a kind of NLU confidence level. In the case of a single value, the CL is simply the probability retrieved by the inference process. When there are several values, an individual CL can be affected to each retrieved AV pair or a global CL for the utterance can be affected by multiplying all individual probabilities. Of course, this provides CL on the attributes while the ASR model provides CL on the values.

It should also be remembered that the language model loaded for speech recognition is state dependent. That means that the SDS allows only some specific spoken entries in a given state. That should be taken into account when applying the NLU process on ASR outputs. This is realized by avoiding entering some evidence in the inference engine. For example, when the system's asks for explicit confirmation, evidence like expressed in the first and the third lines of Table 10 shouldn't be used unless the language model allows

any entry. This also produce misunderstanding errors or null intention outputs when the user has been too over informative or was not cooperative.

VI.2. Output Processing Simulation

Like exposed in sections II.3.3 and II.3.4, the outputs processing block is composed of a NLG and a TTS sub-systems (Fig. 46). The modelling of these subsystems should lead to the estimation of the probability $P(sys_t \mid a_{u,t})$according to equation IV.17. This probability can be expressed as:

$$P(sys_t \mid a_{u,t}) = \underbrace{P(sys_t \mid w_t)}_{TTS} \cdot \underbrace{P(w_t \mid a_{u,t})}_{NLU} \quad \text{(VI.37)}$$

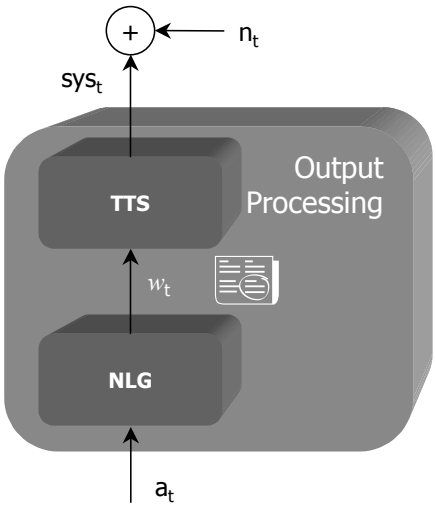


Fig. 46: Outputs processing blocks

VI.2.1. NLG System Model

The NLG block's job can be seen as the translation of the Dialogue Manager's intended action a_t in a sequence of words w_t that communicate the Speech Acts embedded in a_t. As said in section II.3.3, several methods are generally used going from the recorded spoken utterances, handwritten prompts or automatically generated sentences.

Although most of systems use either recorded prompts or more generally human authored prompts (subsequently transformed into speech thanks to a TTS system), the possibility of generating more natural sentences thanks to a completely automatic NLG system is a novel field of researches and even more recent in the framework of SDS [Walker *et al*, 2002]. Indeed, it has been a little more studied in fields of automatic summarisation or document generation [Reiter & Dale, 2000]. Nevertheless, it can be considered that recorded prompts or human authored prompts are not subject to error in the transmission of the concepts while the NLG method might be error prone.

Similarly to the ASR and the NLU techniques, the same conclusion can be drawn for the NLG and the TTS processes. Researches in the field of TTS are those with the longer

history in the field on speech processing and some standard methods have been developed. In the field of NLG, no commonly admitted standard really exist (there are rule-based, stochastic or corpus based NLG methods) and it is therefore difficult to simulate thanks to a prior knowledge of the implementation. Yet, whatever the NLG system, errors in the transmission of the concepts embedded in the systems action a_t generally arise because of bad references in the pronominalisation or the anaphora generation. That means that they can only arise when talking about an attribute that has already been referenced in the conversation. This is why it is proposed to add an ambiguity parameter $\xi_t \in [0,1]$ modifying the meaning of a system's utterance sys_t when a system's action is modified by an already referenced attribute. A confirmation of attribute a^i might be mismatched with a confirmation of attribute a^j if both a^i and a^j have been asked before. This implies ξ_t is not null and the action a_t to be possibly mismatched with another action a'_t (Fig. 47).

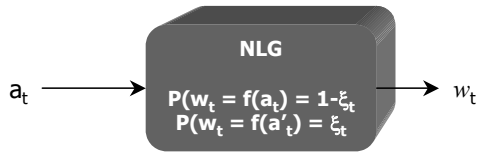


Fig. 47: Error-prone NLG

Of course, this mismatch is generally the result of a misunderstanding of the user's but it is because of an ambiguity in the generated sentence, this is why this parameter is included in a NLG model and not in the user model. The more ξ_t is close to 0, the more the NLG system is accurate and we will consider that ξ_t is null for recorded and human authored prompts.

VI.2.2. TTS System Model

In general, the TTS synthesis process in the framework of a SDS is deterministic in that sense that a given sequence of words w_t will always result in the same spoken utterance sys_t that will always be interpreted in the same way by a given user as a given set of AV pairs. Thus, the only things that can be modelled in the case of a TTS system are objective metrics.

Objective metrics that can be provided by a TTS system are not very numerous in the purpose of a SDS design since it only affects the subjective satisfaction of the user [Walker *et al*, 2001] (hopefully not his/her understanding of the sentence or, it should be avoid to use the TTS system). Indeed, satisfaction surveys showed that the TTS performance was an important factor. Therefore it is proposed to use information about the general perceived quality of the TTS in the computation to generate a metric provided by the TTS model. This can result in decreasing the number of prompts when learning a strategy for example.

On another hand, it has also been shown that the time duration of prompts influences the user's satisfaction. Thus, this duration should also be part of the metrics provided by the TTS model.

VI.3. Conclusion

In this chapter, a simulation model of the whole speech-processing channel has been proposed. It is as least parametric as possible and particularly relies on the specific task.

Indeed, the ASR model takes as inputs the possible values of each attribute in the AVM representation of the task in order to estimate the possible confusion between those values when they are spoken and to provide metrics about the confidence an ASR system can have in the recognition result provided. The inter-word distance computed in the framework of the ASR modelling section of this text can also be used as source of assistance during the design of speech grammars and vocabularies. Indeed it can point out very close words in a given vocabulary or provide information such as the mean inter-word distance or the mean size of a cluster of words in this vocabulary. The parameters of the ASR model are mainly included in the edit cost matrix used for computing inter-word distances. These parameters can be either deduced from articulatory features but also replaced by a measured confusion matrix if a sufficiently large corpus of acoustic and annotated data is available, which is rarely the case for non-expert users.

The NLU model relies on the task structure and on the Bayesian UM already described before and therefore does not require new parameters. It supplies a way to simulate attribute classification errors taking the context into account and provides an estimate of the semantic and contextual confidence level.

The outputs processing sub-systems (namely the NLG and the TTS systems) have also been the subject of a concise modelling study. Possible ambiguities in the generated sentence has been introduced and some metrics about the performance of the TTS systems (specially the time duration of the generated spoken utterances).

In the beginning of this chapter, the noise was also introduced as a variable influencing the process. Several experiments have been realized in order to evaluate the effect of artificially added noise in recorded speech utterance on the confusability between phonemes. Yet, as indicated before, the results obtained on the BDsons database were dramatically bad and nothing could be done to improve them in a reasonable amount of time (since the subject of this thesis was not pure ASR researches). Adding noise to the spoken utterances resulted in a WER close to 100% and no conclusion could reasonably be drawn from these.

Third Part

~

LEARNING STRATEGIES

Children learn to speak very early. Yet smartly interacting with others is a complex and almost lifelong quest mainly based on a trial-and-error process. The definition of a smart interaction is not clear anyway and should be revised for each particular case and each particular participant. No one can actually provide an example of what would have objectively been the perfect sequencing of exchanges after having participate to a dialogue. Human being has a greater propensity to criticise what is wrong than to provide positive proposals.

It then came like obviousness that machines should mimic the natural human behaviour and use unsupervised learning techniques to become able to interact with human beings in a useful and satisfying manner. Moreover, unlike for other artificial intelligence problems where humans perform better than computers, human-authored dialogue strategies are generally sub-optimal because optimality is also a matter of technical performance. The problem is then not only to perform automatic design of strategies but also to outperform human authoring in this domain.

Chapter VII:
Dialogue in the MDP Framework

VII.1. Generalities

From the beginning of this text, a spoken dialogue has been considered as a turn taking process between two agents. It is therefore a sequential process during which two agents try to achieve a goal (generally accomplishing a task: task-oriented dialogue). In order to achieve the agents' mutual goal, they exchange utterances embedding intentions until the task is completed or one of both agents gives up. Although it is not realistic to think about an optimality criterion for generic human-human dialogue, in the special case of task-oriented and goal-directed dialogues, there should be possible to rank interactions and the underlying strategy followed by each agent. It should therefore be possible to learn an optimal strategy according to this optimality criterion. Yet, it is not always easy to identify important indices of dialogue goodness and to derive such a criterion. Moreover, obtaining training data for the special purpose of task-oriented dialogue strategy learning is a very complex problem that should also be optimised.

VII.1.1. Choice of the Learning Method

Learning Strategies

Nowadays, thanks to more than half a century of researches in the field of Artificial Intelligence, lots of learning techniques have been developed. Several problems are addressed by the paradigm of learning algorithms such as pattern matching, decision-making and others, which have as common points that no analytic solution can easily be found and human beings perform better than computers.

Supervised vs. Unsupervised Learning

Two main classes of learning methods can be distinguished: supervised and unsupervised learning. Supervised learning is mainly related to 'learning-by-example' techniques, which means that, during the training phase, examples of a problem are presented to a learning agent as well as the solution. The learning agent then learns to provide good solutions to unseen similar problems. Neural networks are such learning agents widely used for pattern matching (speech or handwriting recognition) [Bishop, 1995].

Unsupervised learning can be seen as 'trial-and-error' learning, which means that the learner is never told what is the optimal behaviour to adopt but is rewarded or punished for its actions in given situations. This last class of learning methods seems to be more natural since it is the way animals and young children learn. Reinforcement Learning (RL) agents are such learners that make use of unsupervised techniques to solve particular decision-making problems [Sutton & Barto, 1998].

In the case of dialogue strategy learning, there is usually no example of optimal strategies available, basically because it is generally unknown for all cases but the simplest ones. Unsupervised learning is then the more suitable technique for this purpose. As a matter of fact, RL has already been proposed to solve the problem of finding optimal dialogue strategies in [Levin *et al*, 1997] and [Singh *et al*, 1999] and proved to be appropriate.

Online, Model-Based, Model-Free Learning

RL has actually been developed in the framework of online learning. As exposed in section III.2 the RL agent interacts with a real environment and learns to behave optimally thanks to rewards obtained while interacting. Thanks to Q-learning algorithms for instance, the RL agent can learn an optimal policy while following another and thus have a coherent behaviour (even if sub-optimal) during learning. Yet, online learning is not really adapted to dialogue strategy design unless a pretty good strategy is already used. Indeed, RL algorithms need a large number of interactions (growing with the size of the state space and the action set) to converge and online learning would involve the system to interact with real human users. This is not thinkable because of the time it would take and the money it would cost.

On another hand, some RL algorithms use a model of the Markov Decision Process (MDP) underlying the studied environment to speed up the convergence. Learning relies on a probabilistic model including the state-transition probabilities $\mathcal{T}$ and the reward distribution $\mathcal{R}$ of the observed environment (see section III.2 for notational information). While interacting, the agent learns from real rewards and updates its model to learn from artificially generated situations between two real interactions. This algorithm family is called *model-based* learning. The most popular model-based RL method is Sutton's Dina-Q algorithm [Sutton & Barto, 1998]. Starting from this point, some researches reported results obtained by learning the state-transition probabilities and reward distribution from a corpus of annotated dialogues and applying a Q-learning RL algorithm on the model [Singh *et al*, 1999] [Walker, 2000]. There are several drawbacks to this technique since it necessitates the collection of a large amount of data but also to predefine all the possible states before collecting the data. Moreover, the data collection only provides information about what happened when using a particular strategy (the one used during the data collection). There might be a lot of states that are not sufficiently visited and above all, this makes the assumption that the strategy does not affect the user's behaviour and that he/she will always act in the same way in a given state whatever the path followed before. This assumption is never met in practice. Finally, it cannot predict what will happen if actions are added to the action set.

For those reasons, the *model-free* technique is proposed. Actually, It is commonly called model-free technique because no prior knowledge about the distribution of states and rewards are needed and above all because the learner doesn't updates its internal model like in the Dyna-Q algorithm. But it actually relies on a model of the environment like described in the preceding three chapters. The thing is that the learner is not 'aware' that it is interacting with a model and not with a real environment; both can be exchanged without any modification in the learner's behaviour. In order to keep things clear, we will

refer to our model-free technique as a *simulation-based* technique. A first simulation-based RL experiment has been described in [Levin *et al*, 1997] followed by [Scheffler & Young, 2002] with the limitations described in section IV.2.1. The subject of this chapter is the application of the RL paradigm to the problem of dialogue strategy learning using the dialogue simulation environment described before.

MDP vs POMDP

As said in section III.2.1, a dialogue can be described either within the MDP framework or within the POMDP framework. Exact solution to POMDP is intractable when the state space is larger than 15 states but approximate solutions that outperform the general MDP solution might be derived more easily. Anyway, the state space must stay very small what is a major drawback for more complex dialogues. The dialogue simulation environment described in the preceding chapters allows the production of data also suitable for POMDP training since it can produce error-prone observed intentions and the corresponding actually emitted intentions. An interesting experiment might be done using some recently developed techniques for solving POMDP my means of the combination of RL and DBN [Sallans, 2002] using dialogue simulation.

Yet, in the following, only the MDP approximation will be described since the rise of performance doesn't justify the extra amount of computation induced by POMDP solving. Moreover, it is definitely not suitable for the purpose of easy design of SDS prototypes. Some tricks will be used to overcome the limitation brought by the MDP approximation and particularly introducing indices of the dialogue system's lacks in observations.

VII.1.2. Preliminary Considerations about SDS Design

In section II.4.1 and II.4.2 were introduced the notions of dialogue model and of dialogue management method. Designing a SDS starts by making strategic choices about those notions that will influence all the design process.

Dialogue Model

In the framework of strategy learning, the dialogue grammar model is not suitable since the RL algorithms purpose is to map particular situations to actions and not to derive general rules about what to do in generic cases. Other models like plan-based or conversational game theory are maybe more suitable but still does not consider the dialogue as a whole. Each turn is considered to be part of a plan but no attempt to repair sub-dialogue explanation is done. Thus, from now on, the joint-activity point of view will be adopted. From this viewpoint, the designer tries to optimise the overall dialogue taking into account possible misunderstanding, problems with the DSP and NLP sub-systems etc. The resulting SDS might then be an agent attempting to optimise each dialogue as a whole according to step-by-step observations.

Dialogue Management

When describing the general RL framework, it has been shown that an MDP could be described as a state-transition network. For this reason but also because of all other reasons exposed in section II.4.2 (like comprehensibility, ease of visual representation,

etc) this dialogue management method will be adopted despite its known drawbacks. It is anyway the only suitable method for MDP representation of a dialogue.

VII.2. Dialogue as an MDP

Applying RL methods to the problem of optimal dialogue strategy learning requires defining a dialogue as an MDP, which means in terms of states, actions, rewards and strategy [Levin *et al*, 1997]. While actions are definitely specific to the SDS being considered and defined by its designer, the state space and the reward are also dependent of the environment. Indeed, the reward is a numerical value provided by the environment and the state is built thanks to the observation coming out of the input processing blocks and processed by the task model.

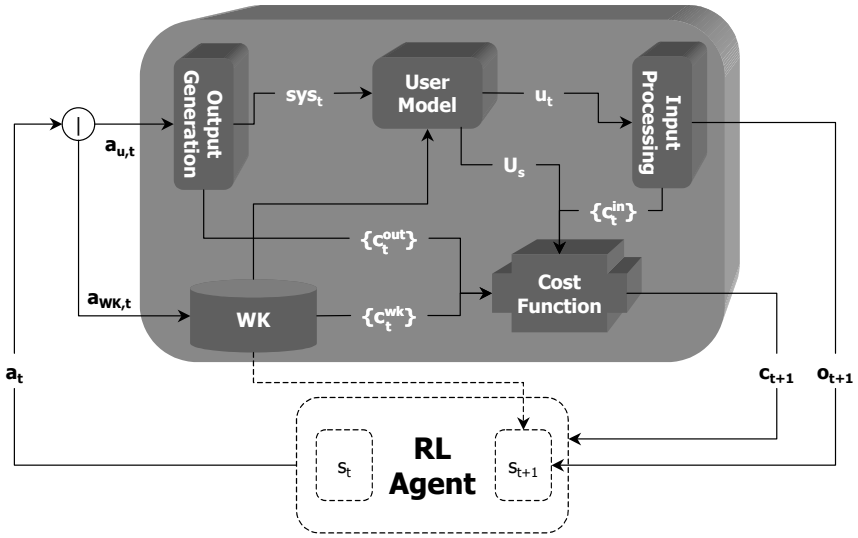


Fig. 48: Learning Process

VII.2.1. State Space

There are of course many ways to build a state space and it is often argued that the state space is very task-dependent. Yet, some general considerations are taken into account:

1. The main one is that each state representation should contain enough information about the history of the dialogue so as the Markov property can be met (see section III.2.1).
2. On another hand, state spaces are often considered as *informational* [Denecke, 2000] in that sense that they are built thanks to the amount of information the DM could retrieve from the environment until it reached the current state.
3. The state representation must embed enough information so as to give an accurate representation of the situation to which an action has to be associated (it is not as obvious as it sounds).

4. The state space must be kept as small as possible since the RL algorithms converge in linear time with the number of states of the underlying MDP.

In the framework of this thesis, several pieces of information are available to build the state space. First, the AVM task representation is known. Second, the output processing blocks provide observations (Fig. 48). Third, the UM might supply information about user's state and level of expertise. Indeed, UM parameters might also be part of the task model helping to build the state from the observations like said in section II.3.5. It can actually be another model or the copy of the one used for simulation purpose but it is not physically the same (not the same instance), this is why it has not been included in the WK on Fig. 48. Yet, in this work, the same architecture will be used for both.

Factored State Space Representation

A factored state space representation has been chosen because, despite it is not generally defined this way, state representations of dialogue systems are often based on state variables. Like exposed in section III.2.1, a factored state space is defined as $S = \{S^{\alpha}\}_{\alpha=1}^{M}$.

In this framework, several state variables have to be defined as well as their respective possible values:

- State variables that are relative to the task. Actually, each attribute $a^i \in A_T$ of the AVM task representation can be a state variable. Values are affected to those variables thanks to recognition results embedded in the observations provided by the environment. In general possible values can either be all the actual possible values given by the AVM or they can be grouped. Typically, those variables can only take 2 or 3 values according to their status: *known*, *unknown* (*confirmed*). This meets the statement 2 in the above.
- State variables $\{m^i\}$ that are relative to the metrics provided by the output processing blocks. For example, the different confidence levels might modify the state. Indeed, when speech recognition provided bad confidence level it might be useful to ask for confirmation. Adding information about system's performance is thus important to meet statement 3 in the above. These variables can take continuous values, discrete values or more generally are values classified as *low*, (*medium*) and *high*.
- State variables relative to environment $\{e^i\}$ itself, like the UM for example. This will be discussed later.

A particular state is then a tuple of state variables and their corresponding values. This tuple is built thanks to output observations and the task model.

Variable type	High granularity	Medium granularity	Low granularity
Task: a^i	$\{v^i\}$	grouped	*Known*, *unknown*, (*confirmed)*
Metrics: m^i	continuous	discrete	*low*, (*medium*), *high*
Env.: e^i	…	…	…

Table 11: Possible state variables and their values

Table 11 shows the different types of state variables and their possible value sets. The larger the value set is, the larger the state space is of course. When the state space is not continuous, the evaluation of the policy requires to keep parameters for each state-action pair. That is the needed memory is proportional to the product of the states space size and the action set size. When using value sets of the right column, each state stands actually for several states obtained by using previous columns. Indeed, in order to reduce the state space, it is common to use *state tying,* which corresponds to make several states share the same parameters (state value, action value, state transition probability and reward distribution).

State tying can actually be used for speeding up convergence of learning algorithms. If enough memory is available, one can build a system based on two different state spaces: a large one *L* and a smaller one *S*. Each state s_S in the smaller state space corresponds to set of states s_L in the larger one which parameters are updated thanks to a back-off rule:

$$\text{param}(s_L)_{t+1} = \lambda \cdot \text{param}(s_S)_t + (1-\lambda) \cdot \text{param}(s_L)_t \qquad \text{(VII.1)}$$

In this expression, $\lambda \in [0,1]$ is a parameter decreasing as the number of visits of state s_L increases. Since the state set *S* is smaller, states s_S are visited more often than states s_L and the parameters of s_L converge more quickly using this backing off technique. Yet it is not always easy to find out which states can share the same parameters (at least during the first learning phase). State tying could be advantageously used in the framework of dialogue strategy learning since we can feel that lots of states are similar. It might be interesting for example to use inductive logic programming to learn those similarities like have been done in [Lecoeuche, 2001].

VII.2.2. Action Set

The action set defines the set of all possible actions the final SDS will be allowed to perform. Different granularity levels can be considered inside the action set. Indeed, elementary actions could be implemented like generating a prompt, loading a speech grammar, starting the speech recogniser etc. On another hand starting a dialogue and let it follow a fixed strategy until the end of the interaction is also an action. Something in between would probably fit better to the need of an easier design. In general, several elemental actions are commonly executed in a fixed sequence not requiring to be optimised. As previously said (see section IV.2.1), two main action families can be distinguished according to which part of the environment they target: the user or the WK. A sequence of elemental actions addressed to a user usually starts with a prompt. The remainder of the sequence is generally motivated by the Speech Acts embedded in the prompt. For example, querying the user for some information by a prompt is followed by loading a grammar in the ASR system, waiting for the answer, processing it by the NLU sub-system and computing the new state according to the observation. Since these sequences are fixed, they can be considered as a unique action in the framework of strategy learning. It is the job of the designer to specify all the possible actions but there are generic actions shared by lots of dialogue system.

Actions addressed to the user are generally system prompts embedding different intentions (Speech Acts or Dialogue Acts) like the following ones:

- Greeting: it is the general definition by the system of its capabilities generally followed by an open-ended question. This action involves loading a large vocabulary grammar in the ASR system in order to

authorise very natural language entries after the prompt and waiting for a speech input.

> **System:** '*Welcome to our train ticket booking service. How may I help you?*'

- Constraining question (system initiative): it is a system-directed action querying the user for a specific piece of information and doesn't allow over-informative behaviour from the user, thus the loaded ASR grammar is often restricted.

> **System:** '*Please, say your departure city.*'

- Open-ended question (mixed initiative): the system asks for more information to the user without specifying strict constraints about the content of the answer.

> **System:** '*How may I help you?*'

- Explicit confirmation (system initiative): the system asks to confirm a specific value before going on. This action also involves loading a very simple grammar allowing for a small set of answers (typically 'yes' or 'no').

> **System:** '*Did you say you want to go to Namur?*'

- Implicit confirmation (mixed initiative): the system asks for confirmation about a specific variable by inserting the previously retrieved value in a query for further information. This relies on the user instinctively contradicting any incorrect information.

> **System:** '*When do you want to go to Namur?*'

- Final confirmation: generally, before ending a dialogue or a sub-dialogue, the system can ask to confirm all variables obtained so far.

> **System:** '*Please confirm the following transaction: you asked for a one-way ticket to go from Namur to Mons on next Friday.*'

- Relaxing prompt: when a specific value involves troubles in the achievement of the task, the system can ask to relax the constraint on the value. It is the case when the system is dedicated to database querying and the provided values results in an empty query result.

> **System:** '*There is no train going from Namur to Mons on next Friday. Would you like to change the departure date?*'

- Repair sub-dialogue: sometimes, the dialogue goes wrong because the user's answers are not coherent or because there has been recognition or understanding errors. The system can then engage in a repair sub-dialogue. Such a sub-dialogue can be composed of a single prompt helping the user to answer correctly to the last query or can be more complex in order to ensure that both the user and the system are sharing the same beliefs (grounding).

> **System:** 'The departure date should be comprised between tomorrow and the 1st of January 2005'

- Assertion prompt: the system transmits information to the user in order to update his/her knowledge. No particular user input is generally expected after an assertion prompt. It might be a prompt aiming at helping people to use the system or to provide information the user asked for.

  System: '*The train you asked for will arrive at line 1.*'

- Dialogue closing: at the end of the dialogue session the system utters an exit prompt and definitively close the session. The user cannot interact anymore unless he/she starts a new session from the beginning.

  System: '*Your ticket will be sent to you by postal mail. Thank you for using our automated booking service.*'

Actions addressed to the WK are typically database queries or accesses to expert systems. Once again they are composed of more basic actions executed in sequences like building the database query, execute the query, retrieve results and build a new state according to those results.

Factored Action Set Representation

Like for the state space, the action set can be represented in a factored manner. This factored representation has already been implicitly introduced with the Bayesian UM in section V.3. In the framework of this thesis, two types of action variables are proposed: one variable (A_S) describing the generic action type (like those just described in the above) and a set of Boolean variables $\{s^i\}$ indicating whether an attribute modifies the action or not. Table 12 shows the different possible values for action variables.

Other action types might be added according to the need of the task but such task-specific design is out of the scope of this thesis.

Variables	Possible Values	Description
Type: A_S	greet	Greeting
	constQ	Constraining question
	openQ	Open-ended question
	explC	Explicit confirmation
	implC	Implicit confirmation
	allC	Confirm all (final)
	rel	Relaxing Prompt
	assert	Assertion prompts
	close	Dialogue closing
	dbQ	Database Query
Modifying Attribute : s^i	Boolean	Indicates whether attribute s^i modifies the action A_s

Table 12: Factored Action Set

In the examples of section V.2, an action has also been denoted as a function like const(dest), for instance. It actually is the same as if the action was described by the tuple $\{A_S = constQ, dest = 1, dep = 0\}$. The 'function' notation is sometimes more convenient for conciseness purpose but it always has a factored counterpart.

VII.2.3. Reward Function

The construction of the reward function is, as much as the state space representation choice, a very tricky job. It is generally very hard to tell what was a good dialogue since it would actually necessitate comparing dialogues with each other. The better metric to evaluate an SDS would probably be user satisfaction. Yet, even this assumption is subject of discussion. Indeed, in the case of a system aiming at a selling task, the better evaluation of the dialogue system would probably be the sales performances, and the system's owner satisfaction instead of pure user satisfaction. Nevertheless, user satisfaction has been widely considered as the better metric of SDS goodness. This is probably motivated by the fact that, if one wants in a close future have the opportunity of evaluating SDS performances according to sales performances, SDS should before be sufficiently attractive to users.

User satisfaction surveys have therefore been widely used in order to assess dialogue systems. Yet, it has been pointed out that surveys aiming at comparing different dialogue systems instead of pure subjective evaluation of a unique system gave results that were hard to use in practice. Indeed, users' evaluations were proved to be dependent of the order in which systems were presented to them [Devillers & Bonneau-Maynard, 1998].

User satisfaction is in general not only dependent of system's performance but it is also very task-dependent. Users' ranking of dialogues are generally done in a very subjective manner. Yet it is a sought-after to find out a way to evaluate dialogue sessions thanks to objective metrics.

Arbitrary choice of the reward function has been proposed in [Scheffler & Young, 2002], where it was defined as cost:

$$C_d = N_t + \text{FailCost} \cdot N_{\text{Failures}} \tag{VII.2}$$

In this equation, C_d stands for the dialogue cost, N_t stands for the number of user turns and $N_{Failures}$ is the 'average number of failures per transaction' representing a mean task completion measure. FailCost is varying a weight indicating the relative importance of failures.

On another hand, as firstly presented in early work on the subject [Levin *et al*, 1997], a better cost function would be obtained using a weighted sum of objective metrics:

$$C_d = \sum_i w_i c_i \tag{VII.3}$$

In this expression, c_i are objective metrics and w_i are weights. In the researches reported in [Levin *et al*, 1997] and [Levin *et al*, 2000], a database querying system is described and metrics used were the number of user turns, the number of tuples retrieved by database queries and the amount of data presented to the user at the end of each dialogue. Those metrics have the big advantage to be easily measurable.

The PARADISE paradigm (see section II.4.5) has been proposed to predict user satisfaction thanks objective metrics [Walker *et al*, 1997a]. The reward function is then expressed as a utility function (not as a cost anymore), which is a weighted sum of objective metrics and of a measure of task completion:

$$U_S = \alpha \cdot N(\kappa) - \sum_i w_i \cdot N(c_i) \tag{VII.4}$$

In (VII.4), U_S is the predicted user satisfaction or the utility function of the studied SDS. Several objective metrics were measured in a wide range of applications and a multiple linear regression was used to compute the weights α and w_i. Metrics used were mean recognition error rate, number of turns, elapsed time, timeout prompts, the number of barge-ins, the number of ASR rejections and the number of help requests. This experiment leaded to lots of different conclusions like explained in section II.4.5. For example [Walker *et al*, 1998 a] reports utility function for two different SDSs, (TOOT and ELVIS) and more recently [Sneele & Waals, 2003] reports results obtained on a Deutch flight information system (IRENE). Table 13 shows a summary of results the reported in those papers. Results of the third column have been obtained by combining results of tests realised with the TOOT and ELVIS systems.

	TOOT	ELVIS	TOOT+ELVIS	IRENE
κ	.45	.20	.23	.39
ASR Perf	.35	.45	.43	.33
Elapsed time		-.23	-.21	
Barge-ins	.42			
User Turns				-.32

Table 13: Objective metrics weights for several SDSs

Except from the first column, it can be concluded that task completion, recognition scores and the number of turns are significant predictors of user satisfaction (number of turns is highly correlated with elapsed time). The second and third columns show that the ASR performances are more important than other metrics while the last column shows approximately equal weights for all metrics. Yet, it is common to all experiments that task completion and ASR performances are significant as well as the number of turns. This is not surprising but some researches reported contradictory results. Actually, as already noticed in section II.4.5 the nature of the survey itself might induce those results [Larsen, 2003], which finally seems also arbitrary. Moreover, these results have been obtained under a particular strategy and the predicted user satisfaction is only valid to evaluate dialogues obtained by following this strategy. Finally, (VII.4) implicitly assumes that measured metrics are independent from each other. This assumption is clearly not met like shown in [Larsen, 2003].

In section V.3.4, a user satisfaction model has been proposed. It was obtained thanks to comparison of the interaction's history with an expected history, number of turns and task completion and had different distribution according to the level of expertise of the model. A set of dialogues has been simulated with this UM on the train ticket booking and the computer-retailing examples following a random strategy. Users with three different levels of expertise were simulated and 1000 dialogues were generated with each of them. Linear regression has then been applied to the results to find out weights related to number of turns, ASR performances and task completion. The results can be found in Table 14 in which TC stands for Task Completion. It is different from the kappa coefficient since it is more related to a perceived task completion measure (see section V.3.4.)

	Low (.1)	Medium (.3)	High (.5)
TC	.31	.32	.31
ASR Perf.	.35	.41	.49
User Turns	-.22	-.31	-.43

Table 14: Weights obtained with simulated user

Results found with the UM are hopefully similar to those obtained previously. Of course it is due to the way user satisfaction has been defined in section V.3.4. Anyway what is important in the framework of dialogue strategy comparison is the relative importance of weights and their evolution with user's level of initiative. Different objective functions can be obtained for different user populations and after all, the user satisfaction described in section V.3.4 could be used.

Yet user satisfaction obtained by using the UM is only available at the end of the interaction, which would slow down the learning process. Moreover, the ASR performance is not an available measure unless the dialogues are logged and manually analysed in order to identify recognition errors. That makes online learning impossible. Using the dialogue simulation environment one could access the number of simulated ASR errors but it would be a trick. Instead, it is proposed to use confidence levels in the reward function and a per-turn reward [Pietquin & Renals, 2002], [Pietquin & Dutoit, 2002] in order to speed up the learning process.

Doing so, the base of the reward function used in the following will be:

$$C_d = w_t \cdot N_t - w_{CL} \cdot CL - w_{TC} \cdot TC \tag{VII.5}$$

with:

- $N_t = 0$ if the dialogue is closed and 1 otherwise,
- CL is the confidence level of the last retrieved utterance,
- TC is a measure of task completion,
- w_i are positive weights set according to Table 14.

An optimal strategy would minimize the mean dialogue cost:

$$\overline{C}_d = E[C_d] = w_t \cdot E[N_t] - w_{CL} \cdot E[CL] - w_{TC} \cdot E[TC] \tag{VII.6}$$

The use of the confidence level instead of the ASR actual performance is motivated by the correlation existing between those two metrics. Indeed, it is the purpose of the CL to reflect ASR performances. Moreover, optimising the dialogue partially according to the CL will conduct to maximize the overall dialogue CL. Since the dialogue simulation environment can provide estimates of the CL it is suitable for learning with this cost function. Yet, the environment can provide other metrics (like the size of the cluster to which belongs the last recognised value, the mean inter-word distance of the vocabulary used to recognise the last value etc.) but those metrics are not naturally provided by real systems. Thus, in order to make the learning system able to learn online without changes, only metrics provided by usual DSP and NLP systems will be used. This way, the simulated environment can be substituted with a real environment or another simulation environment like the one described in [López-Cózar *et al*, 2003] for instance.

Readers will notice that the use of metrics describing confidence in the processing of speech inputs in the state space and in the cost function is also an attempt to overcome the problem involved by partial observability of the process without using the POMDP framework.

VII.3. RL Algorithm and Parameters

The section III.2.1 presented several different algorithms to solve the RL problems. In researches reported in [Pietquin & Renals, 2002], the chosen algorithm was a Monte Carlo method with a ε-greedy action selection strategy. The ε parameter was set close to 1. These choices were motivated by the fact that the learning agent interacts with a simulated environment and so, is in a pure learning process. Thus it doesn't have to follow any consistent strategy but has to evaluate all the state-actions pairs as fast and as many times as possible. This algorithm converged after 20,000 to 70,000 simulated dialogues (this number justifies the use of simulation, at least for early design stages). Yet, those choices have several drawbacks.

First, the learning agent waits the end of a complete dialogue session to update the action values of the visited states. The algorithm convergence is therefore slowed down. Second only the followed policy is evaluated while algorithms like Q-learning enables the learner to evaluate the optimal policy while following another (there actually exist off-policy Monte Carlo learning methods but the problem of late update of action values stands). Finally, the ε-greedy action selection strategy also slows down the learning process since after a certain number of iterations, the learner should give up the evaluation of state-action pairs showing very low Q values.

We finally chose to implement a Watkins Q(λ) algorithm and use a softmax action selection strategy. Moreover, the initial Q values were set to optimistic values. The effect of optimistic initial values is that the learning agent explores a lot at the beginning of the learning process since actions with higher values are selected by the softmax method. The exploration decreases as the learning agent covered sufficiently each state-action pair in order to evaluate them.

The discount rate γ of equation III.22 is set to 1 since the task has a finite horizon, there is no need to use discounted rewards. The learning rate α is set to be dependent of time and is decreasing over time. It was set to 1/k, with k being the number of time the updated state-action pair has been visited. Finally, the λ parameter has to be set. Actually, better results are obtained with very high value of λ (close to 1). Actually, Q(1) is often referred to as a Monte Carlo method while it is not since the Q(1) uses a full back-up at each step and not once at the end of the interaction. Thus, Q(1) converges more quickly than pure averaging Monte Carlo methods described by expression III.15. While it is not suitable for lots of RL problems, it gives better results in this case.

Chapter VIII:
Two Case Studies

VIII.1. Computer-Retailing System

The first example described in the framework of SDS design is the same as described in section V.2: the computer-retailing example. The task mainly consists in a database query system and the goal of the application is to provide the price of a computer selected according to specific features provided by the user. Notice that the complete selling task is not considered (no credit-card number asked or whatever).

The database contains 350 different computer configurations split into 2 tables (for notebooks and desktops) containing 6 fields each: pc_mac (pc or mac), processor_type, processor_speed, ram_size, hdd_size and brand.

VIII.1.1. AVM Task Representation

Since the task relies on an existing database, the AVM representation is quite straightforward. Attributes are the fields of the database and an additional attribute is added for the table name. Generally speaking, choosing the table in a database is often associated to the choice of a goal type. Here, two goal types can be distinguished, buying a notebook or a desktop computer. The AVM representation of this task is shown in Table 15. The used database has three years old so described computers are not sold anymore.

Attributes	#	Values
table	2	'Notebook', 'Desktop'
pc_mac	2	'PC', 'Mac'
processor_type	9	'Pentium II', 'Pentium III', 'Celeron', 'AMD-K6- 2', 'AMD K7', 'AMD Athlon', 'Cyrix', 'G3', 'G4'
processor_speed	551	[350-900] MHz
ram_size	3	32, 64, 128
hdd_size	4	20, 30, 40, 60 Gb
brand	14	'IBM', 'Sony', 'HP', 'Toshiba', etc.

Table 15: AVM representation for the computer retailing task.

Some values for a given attribute exclude values for another (like, PC excludes the 'G3' value for the processor type) but such internal structure will not be specified. Indeed, if the user provides incompatible values, that will lead to an empty result of database queries. The system should learn to behave correctly in such a case.

VIII.1.2. Action Set

Since the task involves database querying, actions will not only imply interaction with the user but also with the database. The following action variables will be considered:

Variable	Values
Type: A_S	'greet', 'constQ', 'openQ', 'expC', 'allC', 'rel', 'dbQ', 'close'
Attributes: table, pc_mac, processor_type, processor_speed, ram_size, hdd_size, brand	{0,1}

Table 16: Action set for the computer-retailing example

Of course, all attributes are not modifying all actions thus, certain values of the A_S variables exclude the 1 value for certain attribute variables. For example, the 'greet' value is not modified by any attribute, neither the 'close' action. On another hand, there are some values involving that only one argument modifies them like the 'cosntQ', 'expC' and 'rel' actions. This reduces dramatically the size of the action set. Finally, there are actions that are modified by all attributes at the same time like 'openQ' (we consider that the open ended questions accepts any attribute or sequence of attributes as a valid answer) and 'allC' and 'dbQ' (since the database query is conditioned by all the attributes provided by the user). According to these considerations, the action set size is 25.

Reader will notice that there is not data presentation action. Actually, it is considered that the data presentation is included in the 'close' action.

VIII.1.3. State Space

As argued previously, the state space is very important and several state spaces can be investigated for the same task. Table 17 shows available state variables for building the current task's state space. In this table, each attribute of the task's AVM is present (Attributes), the 'status' variable indicates whether the attribute has been confirmed by the user or not, each attribute has a corresponding confidence level CL_i and the 'DB' variable indicates the number of database records that correspond to AV pairs retrieved so far. Table 17 also shows that different sets of values can be used for each variable. It is important to know that the larger the value set is, the larger will be the resulting state space.

For example, if one takes the values of the first column to build a state space, its size will reach more than 10^9 states, which would lead to an almost intractable solution (unless using particular techniques like in [Tesauro, 1994]).

When using the value set of the right column (only composed of Boolean values), the state space size decreases to 2^{10} states, which is already tractable but quite large for this simple task.

Variable	Values	
Attributes: table, pc_mac, processor_type, processor_speed, ram_size, hdd_size, brand	Actual Values	{*known*, *unknown*}
Status:		{*confirmed*, *not confirmed*}
ASR CL: {CL_i}	Discrete Value {.0, .1, …, 1.0}	{*high*, *low*}
DB: retrieved records	Actual Values	Number of records or {*high*, *low*}

Table 17: State space variables for the computer-retailing example

The state space size can still be reduced by noticing that when an attribute is '*unknown*', it cannot be '*confirmed*' and has a '*low*' confidence level. Thus some states will never been visited and don't have to be implemented. This reduces the state space size to 2^8.

In the experiment exposed here, two different state spaces are tested. The first is the 2^8-state space using Boolean values described in the above (S_1) and the other, even simpler is realised by not including the 'DB' variable in the set of state variables (S_2).

VIII.1.4. Dialogue Simulation

In the second part of this text, several methods to simulate a dialogue have been proposed. In the present experiments, 3 different configurations were used.

First, a simple simulation (Sim_1) has been realised by connecting a RL agent to a simple probabilistic user model based on a set of probabilities exposed in section V.2.1 and which utterances are filtered by a minimal error modelling with a fixed error rate.

Second, the same simple UM has been connected to the 'Task Classification' error model exposed in section VI.1.1 (Sim_2). Thus, each attribute of the task's AVM has to be categorised and associated with a WER and two CL distributions (one for well recognised word and another for wrong recognitions). The designer should then do this classification manually. The chosen classification is shown in Table 18.

Attributes	#	Recognition Task	WER
table	2	Boolean	.05
pc_mac	2	Boolean	.05
processor_type	9	Isolated words	.2
processor_speed	551	Number	.1
ram_size	3	Number	.1
hdd_size	4	Number	.1
brand	14	Isolated words	.2

Table 18: Recognition task classification for the computer-retailing example

Third, the Bayesian UM has been connected to the automatic ASR model also discussed in section VI.1.1 (Sim_3). The rest of the environment simulation has been disabled for explanatory purposes (in order to compare results in the following).

Finally, this last simulation configuration has been used with a UM presenting 3 different degrees of initiative: low (Sim_{3a}), medium (Sim_{3b}) and high (Sim_{3c}).

For the purpose of simulation, a user goal has to be created before starting each simulated dialogue. As said before, an AVM representation is also used for the goal. Before starting a dialogue session, a goal is so built by filling in a template with a value for each field. A goal example is shown in Table 19.

Attributes	Values
table	'Notebook'
pc_mac	'PC'
processor_type	'Pentium III'
processor_speed	N/A
ram_size	128 Mb
hdd_size	30 Gb
brand	N/A

Table 19: User goal

In order to simulate more realistic behaviour, the user's goal might not contain value for some of the attributes. This suggests that the user doesn't know the value or that he/she doesn't care about the value of a particular attribute (processor_speed and brand in Table 19).

VIII.1.5. Reward Function

Once again, several proposals can be made for building the reward function. Yet, one must remember the general framework defined to do so in section VII.2.3. According to (VII.5), the reward function should rely on an estimate of the number of turns, the confidence level for the last retrieved attributes and the task completion.

Considering the estimate of the number of turns (which actually stands for the total elapsed time), two values are actually available: the number of user turns N_U and the number of system turns N_S (including database queries).

On another hand, the task completion is not always simple to define. The kappa coefficient is one possibility but didn't always prove to correlate well with the perceived task completion. For the purpose of this experiment, two simple task completion measures will be defined:

$$TC_{max} = max\left(\#\left(G_U \cap R\right)\right) \tag{VIII.1}$$

$$TC_{av} = \text{average}\left(\#\left(G_U \cap R\right)\right) \tag{VIII.2}$$

In these last expressions $\#(G_U \cap R)$ is the number of common values in the user's goal G_U and a record R presented to the user at the end of a dialogue. When a value is not present in the user goal (N/A) it is considered as being in common.

The first task completion measure TC_{mas} is an indicator of how close is the closer record in the presented results. The second TC_{av} measures the mean number of common values between the user's goal and each record presented.

VIII.1.6. Results

In the following, results of different experimental settings are reported. Each experiment leads to different learned strategies, which will demonstrate the sensibility to the parameters. Each of the configuration will be denoted by a tuple {*S*, Sim, N, TC) defining the used parameters. In this tuple, *S* defines the state space (S_1 or S_2), Sim indicates the simulation environment configuration (Sim_1, Sim_2, Sim_3), N stands for the time duration measure (N_U or N_S) and TC stands for the task completion measure (TC_{max}, TC_{av}).

Results will be described in terms of average number of turns (user and system), average task completion measure (TC_{max} and TC_{av}) for the performance and in terms of action occurrence frequency during a dialogue session to get a clue about the strategy learned. These results are obtained by simulating 10,000 dialogues with the learned strategy.

First Experiment: {S_2, Sim_1, N_U, TC_{max}}

A first experiment based on the smaller state space (without any clue about the retrieved records) has been realised, with the simplest simulation environment, using the number of user turns as a measure of elapsed time and the TC_{max} as the task completion measure. Results are shown in Table 20.

Performance			
N_U	N_S	TC_{max}	TC_{av}
2.25	3.35	6.7	1.2

Strategy							
greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	0.06	0.0	0.14	0.0	0.05	1.10	1.00

Table 20: Learning results for {S_2, Sim_1, N_U, TC_{max}} configuration

When looking at the three first columns of the performance table, the learned strategy doesn't look so bad. It actually has a short duration in terms of user turns as well as in system turns and has a very high task completion rate in terms of TC_{max} measure. Yet the TC_{av} shows a very low mean value.

When looking to the average frequency of actions in the strategy table, one can see the only action addressed to the user that happens frequently during a dialogue is the greeting action. Others almost never happen. Actually, the learned strategy utters the greeting prompt to which the user should answer by providing some argument values, performs a database query with the retrieved attributes and provides the results to the user. Sometimes, the user doesn't provide any attribute when answering to the greeting prompt or the value is not recognized at all by the ASR model, so the strategy is to perform a constraining question (and not an open ended question) that will provide an argument with a better CL. Sometimes the provided arguments have a poor CL and an explicit confirmation is asked for and sometimes the provided arguments doesn't correspond to any valid record in the database so the strategy is to ask for relaxation of one argument (this also explains why the number of database queries is greater than 1). The reason for which TC_{max} is not maximal (equal 7) is that sometimes, the dialogue failed.

This results in showing to the user almost all the database records when he/she only provides one argument when prompted by the greeting. This is why there is a so big difference between TC_{max} and TC_{av}. The desired record is actually in the presented data

(TC_{max} is high) but it is very difficult to find (TC_{av} is low). This strategy is definitely not suitable for a real system.

Second Experiment: {S_1, Sim_1, N_U, TC_{av}}

This experiment uses the same settings as the previous one except that the 'DB' variable is added to the state variables and the task completion is measured with TC_{av}.

Performance			
N_U	N_S	TC_{max}	TC_{av}
5.75	8.88	6.7	6.2

Strategy							
greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	0.87	1.24	0.31	1.12	0.21	3.13	1.00

Table 21: Learning results for {S_2, Sim_1, N_U, TC_{av}} configuration

This time, TC_{max} and TC_{av} are close to each other, showing that the presented results are more accurate but the number of turns has increased. The number of system turns particularly shows higher values. This observation is obviously explained by the increase of database queries (since it is the only action that increases the number of system turns).

A look at the action occurrence frequencies will help understanding what happened. The learning agent obviously tries to maximize the TC_{av} value while minimizing the number of user turns and maximizing recognition performance. To do so, it always performs a database query after having retrieved information from the user. Since the number of results is part of the state representation, the agent learned not to present the results to the user when in a state with a high number of results. If this number is to high after the greeting, the learner tries to reach a state where the result number is lower. Thus it almost systematically performs an 'openQ' action after the greeting in order to get as much information as possible in a minimum of turns (this explains the 1.24 value). Yet, this often result in poorer recognition outputs, thus it also performs a confirmation of all the fields before presenting any results. Sometimes, more information is provided when the user answers the greeting and only a constraining question is asked to gather enough information so as to reach a state with less results retrieved from the database. A constraining question is preferred in this case because it leads to better recognition results.

The mean number of user turns shows that only 5.2 turns are usually needed to reach an accurate result set because the computer configurations are sufficiently different so as not to need too much attributes in the database query to provided accurate results. Thus, the system doesn't ask for all the attribute values to the user.

Third Experiment: {S_1, Sim_1, N_S, TC_{av}}

The same experiment as the previous one has been performed but replacing the N_U measure of time duration by the N_S measure. It actually make sense since in a real application, the database could be much larger than the one used here. Thus, the database queries could be much more time consuming and waiting would bore the user. Results of this experiment are shown in Table 22. This obviously results in a decrease of the number of database queries involving a proportional decrease of the number of system turns N_S. Yet, an increase of the number of user turns U_S is also observed.

Performance			
N_U	N_S	TC_{max}	TC_{av}
6.77	7.99	6.6	6.1

Strategy							
greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	1.57	1.24	0.33	1.32	0.31	1.22	1.00

Table 22: Learning results for {S_1, Sim_1, N_S, TC_{av}} configuration

By examining the action frequencies, one can notice that the number of constraining questions increased resulting in an increase of N_U. Indeed, the learned strategy implies gathering enough information from the user before performing a database query. This explains why the systems ask more constraining questions.

Fourth Experiment: {S_1, Sim_2, N_S, TC_{av}}

In this experiment, the simulation environment has been modified and the Sim_2 environment has been used. The main change is the ASR model, which includes different error rates and confidence level distributions for the different recognition tasks.

Performance			
N_U	N_S	TC_{max}	TC_{av}
6.53	7.72	6.7	6.2

Strategy							
greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	1.56	1.22	0.21	1.32	0.22	1.19	1.00

Table 23: Learning results for {S_1, Sim_2, N_S, TC_{av}} configuration

Examining the performance indices, one can see a small reduction in the number of turns (N_U and N_S) and a significant reduction in the number of explicit confirmation and relaxation prompts. To understand what happened, another measure should be observed: the actions value in each state. To have a comparative view of different action values in a same state, the softmax probability of each action is a better metric than the actual values. For example, let's examine the softmax probability of each constraining questions in the initial state (when nothing is known yet). Results are shown in Table 24.

note_desk	pc_mac	p_type	p_speed	ram	hdd	brand
Experiment {S_1, Sim_1, N_S, TC_{av}}						
0.1	0.1	0.01	0.04	0.04	0.04	0.01
Experiment {S_1, Sim_2, N_S, TC_{av}}						
0.05	0.05	0.05	0.05	0.05	0.05	0.05

Table 24: Comparison of initial state Q-values

We can see that the probabilities have been modified comparing to the previous experiment (shown on the second result line). Some actions have a greater probability to occur using a softmax action selection strategy, which means a higher value. This is due to the maximisation of the CL value. The learned strategy now orders the constraining questions in order to ask first the values that will produce a better recognition result. In the case of the task classification simulation model, better results are obtained with the 'Boolean' recognition task followed by the 'number' task and finally the 'isolated words' task provides the worst results. This also conducted to a small reduction of the number of database queries since some bad results provided by the recognition error model resulted in empty queries.

Fifth Experiment: {S_1, Sim_3, N_S, TC_{av}}

The last experiment is done thanks to the Sim_3 simulation environment in which the Bayesian UM and the automatic ASR model are implemented. A first experiment has been done with a UM with a low initiative level and results are described in Table 25.

Performance			
N_U	N_S	TC_{max}	TC_{av}
6.54	7.65	6.7	6.3

Strategy							
greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	1.55	1.22	0.20	1.33	0.24	1.21	1.00

Table 25: Learning results for {S_1, Sim_3, N_S, TC_{av}} configuration

Results seem approximately similar to those obtained with the previous Sim_2 but a closer look at the softmax probabilities will show a significant difference:

note_desk	pc_mac	p_type	p_speed	ram	hdd	brand
Experiment {S_1, Sim_1, N_S, TC_{av}}						
0.1	0.1	0.01	0.04	0.04	0.04	0.01
Experiment {S_1, Sim_2, N_S, TC_{av}}						
0.05	0.05	0.05	0.05	0.05	0.05	0.05
Experiment {S_1, Sim_3, N_S, TC_{av}}						
0.14	0.12	0.01	0.01	0.03	0.02	0.01

Table 26: Comparison of initial state Q-values

This time, the order is once more modified according to the actual properties of the used vocabularies. It is quite difficult to draw conclusion thanks to comparisons with the previous experiments since it makes assumption on the actual performance of a real ASR systems. Comparisons should be made with a real system, which as not been possible.

Other experiments have been realized by making varying the UM's level of initiative. Results are shown below.

Performance			
N_U	N_S	TC_{max}	TC_{av}
6.54	7.65	6.7	6.3
5.99	7.23	6.6	5.9
5.74	7.06	6.6	5.2

Strategy							
greet	constQ	openQ	expC	AllC	rel	dbQ	close
1.00	1.55	1.22	0.20	1.33	0.24	1.21	1.00
1.00	0.80	1.67	0.19	1.02	0.31	1.24	1.00
1.00	0.50	1.86	0.15	0.88	0.35	1.32	1.00

Table 27: Learning results for different user initiative levels

Results are quite easily interpreted since the number of system initiative actions decreased, the number of user turns decreased also but the number of system turns decreased not as much because some errors implied to relax constrains and realise more database queries.

VIII.1.7. Conclusion

Different experiments have been realized and showed the importance of the choices made when designing the MDP underlying the dialogue system being developed. It also showed that wrong design of the reward function was not the only responsible for obtaining sub-optimal behaviour but also too simple state space representations. Yet a sensible point is the definition of the task completion measure.

Moreover, optimising the dialogue according to the ASR confidence level resulted in an adaptation of the learned strategy to the recognition task difficulties. It has also been shown that the strategy can be adapted to different user initiative levels.

VIII.2. Train Ticket Booking System

A second simple SDS has been described in section V.2. This example was actually a simplified version of a train ticket booking system that aimed at delivering train tickets corresponding to a particular travel. Users are invited to provide information about the departure city and time as well as the destination city and time. The desired class was also a possible option.

It is a very common type of SDS and lots of implementations exist. As no real system was available for the purpose of this work, this problem has been considered with a theoretical point of view. An artificial application has been built and, from now on, it will be considered as a form-filling problem.

VIII.2.1. AVM Task Representation

This application is actually a pretext to study the form-filling general application. In this case, attributes of the AVM are the slots to be filled in and values are of course the different values that can be taken by each slot. In the case described here, the different slots are the departure city ('dep'), the destination city ('dest'), the departure time ('t_dep'), the desired arrival time ('t_dest') and the class. Fifty values have been artificially chosen in a list of Belgian cities to stand for the departure and destination cities and time will be considered as being plain hours (8 AM, 3 PM etc). The class values are 'business' or 'economic'.

Attributes	#	Values
dep	50	'Namur', 'Bruxelles', 'Mons', 'Charleroi', 'Liège',…
dest	50	'Namur', 'Bruxelles', 'Mons', 'Charleroi', 'Liège',…
t_dep	24	'1 AM', '2 AM', …, '1 PM', '2 PM', …
t_dest	24	'1 AM', '2 AM', …, '1 PM', '2 PM', …
class	2	'Economic', 'Business'

Table 28: AVM representation of the train ticket example

In the following, the simple problem of retrieving the values for each field will be studied. Of course, a real system would be much more complex and database retrieval

should be used to find out which options correspond better to the user's requirements. Yet, this simple example has been designed to illustrate some interesting results.

VIII.2.2. Action Set

As said just before, this task doesn't require database queries but is a pure form-filling problem. Thus, there is no action relative to the WK and they are all addressed to the user. The relaxation prompt and the allC actions will also be suppressed.

Variable	Values
Type: A_S	'greet', 'constQ', 'openQ', 'expC', 'close'
Attributes: dep, dest, t_dep, t_dest, class	{0,1}

Table 29: Action set for the ticket example

Like said for the computer-retailing example, all attributes cannot modify any actions thus, certain values of the A_S variables exclude the 1 value for certain attribute variables. On another hand, there are some values involving that only one argument modifies them like the 'constQ', 'expC' and 'rel' actions. Unlikely, it will be considered that the 'openQ' action will take at least 2 and at most 3 arguments. The 'function' notation will be used to keep results clear in the following. For example, the action openQ(dep, dest) which actually corresponds to the factored representation {A_S = 'openQ', dep = 1, dest = 1, t_dest = 0, t_dep = 0, class = 0} will result in the following prompt:

System: *What are your departure and destination cities?*

Like in the preceding example, it will be assumed that the 'close' action will return the retrieved information to the user at the end of the dialogue session.

VIII.2.3. State Space

Like in the previous example and as described in section VII.2.1, the attributes of the AVM task representation are part of the state variable set. These variable can take either the actual values reported in the AVM or be set as Boolean variables. The status variable is still included in the state variables set and indicates whether an attribute is confirmed or not.

Variable	Values	
Attributes: dep, dest, t_dep, t_dest, class	Actual Values	{*known*, *unknown*}
Status:		{confirmed, not confirmed}
ASR CL: $\{CL_{Ri}\}$	Discrete Value {.0, .1, …, 1.0}	{*high*, *low*}
NLU CL: $\{CL_{Ui}\}$	Discrete Value {.0, .1, …, 1.0}	{*high*, *low*}
ASR-NLU CL: $\{CL_{Ti}\}$	Discrete Value {.0, .1, …, 1.0}	{*high*, *low*}

Table 30: State space variables for the computer-retailing example

Yet, two new variables will be introduced, namely the NLU confidence level (CL_U) and a combination of both NLU and ASR confidence levels (CL_T). The combination is obtained by multiplying the CLs provided by both system models. This implies the use of the more

complex simulation environment in which model of those systems are implemented. The state variables are summarised in Table 30.

Once again, the state variable values described in the left column leads to a very large state space which size would be 1.44 10^9. Thus, the right column variables will be retained and only one of the three proposed CL will be used. This will result in a state space size of 2^6 states since only one confidence level is used at a time.

The state space including the $\{CL_{Ri}\}$ variables will be denoted S_R, the state space including the $\{CL_{Ui}\}$ will be denoted S_U and the state space including the $\{CL_{Ti}\}$ variables will be denoted S_T.

VIII.2.4. Dialogue Simulation

The dialogue simulation used for this experiment is the more complex one described in section V.3 with the Bayesian UM and the automatic ASR and NLU models. Nevertheless, two configurations will be used in which the NLU model is either disabled (Sim_1) or enabled (Sim_2).

A user goal has to be built before each interaction. As usual, the AVM representation is used to build the goal. This time, unlike previously, the user goal always contains a value for each attribute. An example of a user goal is shown in table Table 31.

Attributes	Values
dep	'Mons'
dest	'Namur'
t_dep	'9 AM'
t_dest	'6 PM'
class	'business'

Table 31: User goal

VIII.2.5. Reward Function

To build the reward function, estimates of the duration time, confidence level and task completion are still needed. The duration time will be estimated thanks to the number of user turns N_U since there is no particular action not involving only the user. The confidence level will be estimated either by the CL_R or the CL_T value.

The task completion will be measured by comparing the data presented at the end of each dialogue with the initial user goal. An estimate of the task completion will simply be the proportion of common values (0, 0.2, 0.4, 0.6, 0.8, 1)

The simulation environment will implement a user model with a medium initiative level.

VIII.2.6. Results

In the following, results of different experiment settings are reported. The settings are modified to point out differences in learned strategies according to the presence or the absence of an error prone NLU model. This model is disabled when using the Sim_1 and CL_A settings and is enabled when using Sim_2 and CL_T settings. The state variable set includes or not the provided NLU metrics when using S_A or S_T. The simulation settings will then be referred to as tuple {*S*, Sim, CL}

Results will be described in terms of average number of turns and average task completion measure for the performance and in terms of action occurrence frequency during a dialogue session to get a clue about the strategy learned. These results are obtained by simulating 10,000 dialogues with the learned strategy

First Experiment: {S_A, Sim_1, CL_A }

A first experiment is performed to serve as a baseline example. In this experiment, the NLU model is disabled, which means that no understanding error is introduced. No information about the NLU model's metrics is used neither in the state space representation nor the reward function. The user model is set up with a medium level of initiative.

Performance	
N_U	TC
5.39	0.81

Strategy				
greet	constQ	openQ	expC	close
1.0	0.85	1.23	1.31	1.0

Table 32: Learning results for {S_A, Sim_1, CL_A } configuration

This experiment shows that with a medium initiative level, the user's satisfaction relies as much on the duration time as on the task completion. Thus dialogues are short, but task completion is not optimal since one attribute is often missing in the presented data (one of the cities in general). There are more open-ended questions than constraining questions. Actually, constraining questions are present because sometimes only one argument is missing and there is no need of an open-ended question to retrieve it. Yet, there are explicit confirmations because the task completion is a factor of user satisfaction. It actually illustrate well the trade-off between task completion and time duration.

Second Experiment: {S_A, Sim_2, CL_A }

This second experiment is performed by using the same state space and the same reward function but by enabling the NLU model in the simulation environment. This results in NLU errors not predictable by the confidence level (thus, the explicit confirmation will not be dependent of the value of the CL state variable). Before having a closer look to the results of this experiment, it should be noticed that the baseline strategy learned in the preceding example leads to a task completion of 0.69 when faced to the configuration explained in this section because of NLU errors.

Results of this experiment are shown in Table 33.

Performance	
N_U	TC
7.03	0.74

Strategy				
greet	constQ	openQ	expC	close
1.0	1.25	1.18	2.60	1.0

Table 33: Learning results for {S_A, Sim_2, CL_A } configuration

In this case, the duration time increases and the task completion decreases. Yet, as explained before, the task completion would have been worst if the number of turns didn't increase (using the previously learned strategy). This increase is due to an increase in the number of explicit confirmations explained by the fact that the system must confirm more values since it cannot use the information about the confidence level.

Third Experiment: {S_T, Sim_2, CL_T }

This time, the NLU metrics are used in association with the ASR metrics to build the state space and the reward function. Results of this experiment can be found in Table 34.

Performance	
N_U	TC
5.82	0.79

Strategy				
greet	constQ	openQ	expC	close
1.0	1.05	1.18	1.58	1.0

Table 34: Learning results for {S_T, Sim_2, CL_T } configuration

These results are comparable to those obtained in the baseline experiment. That means that incorporating the NLU metrics when facing NLU errors reaches approximately the same performance than incorporating ASR metrics when facing ASR errors. In the same way that ASR metrics changed the action values, the NLU metrics changed the values of mixed-initiative actions. Like previously, let's have a look to some action values of the initial state.

openQ(dest,dep)	openQ(t_dest,dep)	openQ(dest,t_dep)	openQ(dest,class)
0.005	0.11	0.11	0.12

Table 35: Comparison of initial state Q-values

Table 35 shows softmax probabilities of some actions in the initial state and it can be seen that the mixed-initiative query associating the dest and dep attributes has a lower probability to occur than other actions. It is actually indicating that the confidence level obtained when asking two city names and trying to associate them to their corresponding attributes is lower than when asking two other attributes.

This result is not surprising since the values of the to city attributes dep and dest are the same. Thus, the NLU process is made more difficult and this results in NLU errors and bad NLU confidence levels. Yet, some other actions have a greater value but are not particularly meeting ergonomic rules. For example asking the destination city and the class before all is not an expected behaviour. Anyway, the designer should fix this problem afterwards.

VIII.2.7. Conclusion

The introduction of NLU errors induces problems that are handled by baseline strategies by augmenting the confirmation mechanisms and restricting mixed-initiative behaviour. The introduction of the NLU metrics in the state space and the reward function results in an adaptation hopefully comparable to the one obtained when introducing ASR metrics to handle ASR errors. This motivates the use of the NLU model in the RL paradigm.

VIII.3. User Adaptation and Goal Inference

The possibility to use the UM as a part of the WK has been mentioned in section VII.2. One possible use of UM would be to create several models for different groups of 'same-minded' users (content-based user adaptation) or even one model for each user if the application is to be often used by a same user (history-based adaptation).

In this work, it has been shown that different strategies can be learned for users with different levels of expertise (or initiative). Objective functions are different and factors of user satisfaction are varying. It would then be a certain advantage to detect the initiative level of the current user in order to adapt the strategy instead of applying the same rules for any user. Of course, user adaptation often involves longer scenarios than those discussed so far but it might be interesting to find out a method to make this possible.

The Bayesian UM described in section V.3 allows for probabilistic inference. The level of expertise is a node of the network and since it has no parent, each of its value defines a new sub-network. Finding which sub-network is the more likely to have generated a given utterance is finding the level of expertise of the user. To do so, evidence can be created with the first utterance of the user, for instance. For example, let's consider the train ticket booking service in which the user gives the values of dest and t_dest after the greeting action. The corresponding evidence is shown in Table 36 which refers to notations of section V.3.5.

A_S	k^i	u^{dest}	u^{t_dest}	u^{dep}	u^{t_dep}	u^{class}	VV_{dest}	VV_{t_dest}
'greet'	*low* ∀i	1	1	0	0	0	'Namur'	'1 PM'

Table 36: Evidence for user adaptation

Entering this evidence in the inference engine of the user model will provide the probability distribution over the value of the initiative level node (I). Taking the value with the higher probability do the job of user classification.

This value might be part of the state variables and could modify the action taken. Yet one must not forget that this will multiply the state space size by the number of possible values for the expertise level and increase linearly the time required for convergence. It is therefore probably a better option to train different strategy on particular UMs and an action 'changeStrategy' to the action set which is forced when the strategy followed is not the one trained for the type of user being interacting with the system.

Of course the classification task might not be able to provide good results with the initial answer. Thus, the system can use its history estimate in order to generate new evidence.

For instance, assuming that the user answered a value for the dep variable to a constraining question following the greeting. The history of the systems indicates that the user already provided a value for the dest and t_dest attributes and didn't confirm it. The corresponding k^i are set to *medium* the gv^i values are set to the values provided before (they are now assumed to be part of the user's goal) and the corresponding variables a^i are set to 1. Entering the corresponding evidence in the inference engine provides a probability distribution over the expertise level values.

An experiment have been done by producing dialogues using the same configuration as in the fifth experiment of section VIII.1.6. The user classification has been applied afterwards using the log of the simulated dialogues and results showed a classification rate of 84% after the first utterance and 95% after the second. Of course the task is made easier and results are optimistic because the same model that has been used for classification generated the dialogues but no other experimental data was available.

Yet, the classification algorithm doesn't give 100% of good results because there is always a probability that a particular user model generated a behaviour that is more likely to be produced by another model.

In the same way, goal inference could be realized if the task structure is reflected in the Bayesian UM's goal as explained in section V.3.3. The evidence would then be composed of values for a^i and gv^i and the inference process would concentrate on finding the mode likely value of an attribute a^j defining precisely the goal. The task structure could be learned from data corpus like proposed in [Meng *et al*, 1999] or handcrafted like proposed in [Wai *et al*, 2001].

VIII.4. Grounding

Eventually, a first attempt to introduce grounding in the RL paradigm has been realized. Indeed, it is also possible to use probabilistic inference with the Bayesian UM to retrieve the most probable values of the k^i variables representing the history of the interaction from the user's point of view. To do so, a new experiment setting has been tested. Starting from the train ticket booking example, some constraints have been introduced in the UM. The main constraint is that the UM doesn't provide the departure (arrival) time if it has not provided the departure (arrival) city before. This is traduced by setting some probabilities to very low values in the Bayesian UM such as $P(u^{t\_dest} = 1 | k^{dest} = \text{'}low\text{'}, \ldots.) = 0.01$.

Suppose then that, after the greeting, the system shifts to a state where it seems to have retrieved values for the dep and the t_dest variables. This can be introduced as evidence in the inference engine in order to retrieve the most probable values of the k^i. It will find that this answer occurs more probably when k^{dest} is set to '*medium*' while the system state indicates that no value for dest has been retrieved.

To experiment this, the Bayesian UM has been connected to the simulation environment and removed the NLU confidence level from the state space and reward function. Yet, a new variable has been added to the state variables: a 'grounding confidence level' variable. This variable is set to 0 when a difference has been noticed between the k^i and the state variables corresponding to attributes and to 1 otherwise. Yet, no new action has been added to the action set. In a real system, a 'repair' action should probably be added and correspond to a repair sub-dialogue in which the system would try to confirm values and to retrieve the actually uttered AV pairs.

The experiment leaded to results similar to those obtained by using the NLU model metrics as state variables and including them in the reward function. That is, the systems learned to ask open-ended questions that more often leaded to a 1 value for the 'grounding confidence level' variable, which are open-ended questions associating dep (dest) and t_dep (t_dest) variables. Yet, the behaviour of the UM leaded to an order in the question asked (first cities and time after).

This experiment can be criticised because adding handwritten probabilities to indicate a preference in the order of questions in the UM induces to learn a strategy according to this order. It makes assumptions that are not always true, it forces the system to ask questions in the preferred order because the dialogue lengths is otherwise increased (since the user refuses to answer questions in another order) and the system tends at learning a strategy similar to the one that could be handcrafted. Yet, this setting has been tested for experimental purposes.

Fourth Part

~

CONCLUSION AND FURTHER WORKS

Chapter IX: Conclusions

IX.1. Simulation

In its second part, this text presented a probabilistic framework for task-oriented and goal-directed spoken dialogue simulation based on the statistic modelling of each component surrounding a dialogue manager. The guidelines followed during the conception of this dialogue simulation environment were the desired goal-directed and data-driven behaviour of each component and the ease of portability to task-specific applications. To achieve these aims, models relying on a small set of tuneable or learnable parameters were proposed. Despite only tuned versions of these models were used in this thesis (because of the absence of training data), the possibility to learn parameters from data or even online refinement of tuned parameters are allowed and easily feasible. The absence of training data was actually the starting point of this work.

Two proposals for goal-directed user modelling were suggested, the first being quite simple and relying on a set of probabilities but providing good enough performance for some applications. The second is based on Bayesian networks, which have known a recent interest in a wide range of artificial intelligence applications. Bayesian networks for user modelling in the framework of spoken dialogue simulation (generative approach) is a new application even if they have already been used for encoding user knowledge and goal inferring in few other researches before. The user model is used in several ways during the dialogue simulation and even for learning purposes. The user model is the trickier module to tune since human behaviour is awfully unpredictable. This simulation component is probably the one that more deserves to be trained on data even though hand-tuned models provided pretty good results when used for dialogue strategy learning.

Stochastic models for digital speech processing and natural language processing modules were also proposed. First, a simple automatic speech recognition model based on task classification has been described. It was obtained by measuring performance of a real speech recognition engine on several recognition tasks. It also showed to produce good results when used for simple strategy learning. Subsequently, a data-driven speech recogniser model has been suggested. It offers more flexible behaviour and automatic adaptability to recognition tasks according to their associated vocabularies. It doesn't require particular tuning but parameters (like phoneme a confusion matrix and inter-word

distances) can be either computed thanks to a feature-based distance or learned from data if available. Application of such a speech recognition engine model in the purpose of spoken dialogue system simulation and comparison is also a new proposal. The feature-based inter-word distance developed in this framework can also serve as help in the design of speech recognition vocabularies indicating similarities between words potentially resulting in recognition errors. It could be used to create clusters inside a French vocabulary, isolating similar words.

An attempt to model natural language understanding errors and confidence levels is also supplied. It allows for contextual natural language understanding simulation at some extent. The model relies on the generative Bayesian user model and uses statistical inferring to reproduce understanding behaviour. Since it relies on predefined Bayesian networks, it doesn't need to be parameterised.

Other modules were also added in the dialogue simulation environment like the natural language generation and text-to-speech system models but at a very simpler level. Particularly, an ambiguity in the generated sentences depending on the context has been introduced but has not really been tested.

The proposed simulation environment allows for reproduction of spoken dialogue at an intention level taking into account physical properties of the speech signal to model errors in the transmitted concepts. More than reproducing the modifying behaviour of spoken dialogue systems, it also provides estimates of the metrics such a system would generate such as confidence levels, number of retrieved records in a database or time duration of synthesised spoken utterances.

IX.2. Strategy Learning and Optimisation

Besides other interests of computer-based spoken dialogue simulation like system evaluation or model assessment, the dialogue simulation environment described in the second part of this text has been used for optimal strategy learning. Strategy learning in the framework of dialogue management is actually helpful for the design but also for te refinement of existing system. Indeed, unlike for other artificial intelligence problems where humans perform better than computers, human-authored dialogue strategies are generally sub-optimal because optimality is also a matter of technical performance. Dialogue strategy learning is then more than an automation problem but also a matter of outperforming human-authored solutions. Nevertheless, strategy optimisation by means of unsupervised learning techniques needs a large amount of interactions. For these reasons, simulation is preferred.

The third part of this thesis presented methods to put the dialogue strategy learning problem in the framework of factored Markov Decision Processes which is suitable for applying Reinforcement Learning techniques. Reinforcement Learning is an unsupervised learning technique using trial-and-error methods to find optimal solution to decision-making and optimal control problems. The factored representation, consisting in defining states and actions in terms of state and action variables, made easier the construction of experimental settings. The granularity inside variable value sets also allows easy creation of state spaces and actions sets differing by their size. Indeed, clustering state variables, for instance, leads to state tying techniques speeding up the learning process.

Experiments performed in this work showed that different settings of the simulation environment and the Markov Decision Process parameters produced significant

differences in the learned strategies. Strategies often differ when taking into account the performance of the speech and natural language processing modules being part of a spoken dialogue system. Yet those performances are not always available for learning and especially for online learning. Indeed, a Reinforcement Learning agent is also able to learn online. Even if in the case of spoken dialogue system optimisation simulation is preferred, it is argued that the simulation process should approach as much as possible the real-world behaviour in order to be able to substitute the simulation environment with a real system. In real cases, speech recognition and natural language understanding performances are not known by means of comparison between actually uttered sentences and recognized attribute-value pairs. But speech recognisers and natural language understanding systems might be able to supply information about their confidence in the result. Thus, this confidence levels have been used as part of the objective function of the learning process instead of usually used metrics often known *a posteriori*.

Experimental results showed that optimising dialogue strategies according to both speech recognition and understanding performances made sense and leaded to pretty good results. The learned strategy is adapted to performance and thus leads to greater user satisfaction. Yet, such an automatic process might not easily take decisions about equally valuable actions (in terms of objective performance) in a given state. Unless the user model encodes preferences about the sequencing of actions, the learning agent cannot distinguish those actions. Yet manually encoding preferences in a user model would lead approximately to learn a handcrafted strategy virtually preset in the user model. Thus, it is argued that the optimisation process should not be asked to solve subjective preference problems unless the user model has been learned and not handcrafted.

Intensive utilisation has been made of the Bayesian user model, even in the optimisation process. An attempt at using this user model for user adaptation and goal inference has also been proposed.

Although the automatic learning of dialogue strategy depends on a lot of parameters, it has been shown that experimental settings could easily be obtained by setting user model parameters while speech recognition and understanding models were data-driven and could be automatically tuned.

Chapter X:
Further Works

X.1. Completing the Design

The specifications described in section I.2 involved not only the optimisation of dialogue strategies but the development of a software dedicated to the complete design of dialogue system prototypes. The results presented in this work are only a step toward this goal but are not directly usable by novice designers. Part of the time granted to this work has been assigned to the design of tools easing the use of those results [Pietquin & Dutoit, 2003].

With the growing influence of the Internet and the recent emergence of the XML technology, the W3C consortium defined the VoiceXML markup language to ease the development of SDSs [McGlashan *et al*, 2004]. Indeed, the VoiceXML markup language describes a universal way to express dialogues featuring speech synthesis, automatic speech recognition and so on. Beyond that, the major strength of this language is to dissociate dialogue description from the complex and low level programming generally induced by dialogue system development. This way, the dialogue description stands in a set of scripts whereas the low level programming takes place in a browser that will interpret the scripts and produce the real sequences of interactions. This allows designers without particular skills to access voice technologies. Moreover, VoiceXML technology allows platform-independent and portable design in opposition with previous dialogue systems design tools [McTear, 1998][Klemmer *et al*, 2000].

With the concepts of VoiceXML scripts and browsers, dialogue systems design becomes analogous to web sites design. Yet, most of nowadays amateur web designers don't write HTML code anymore as graphical design tools became widespread during last few years. Thanks to those tools, a large number of amateur designers could draw their web site and brought their contributions to the growth of the Internet. In the same way, in the purpose of easing the dialogue designer's job and to allow amateur designers to create their own dialogues, a graphical interface able to produce VoiceXML scripts was developed.

Therefore, in the aim of enabling non-specialist designers to create their own spoken dialogue systems, it is proposed to use the results of the RL algorithm all along the design process to advise the designer. In this manner, designing a dialogue strategy will become a semi-automatic process involving itself a machine and a human operator.

For the reasons described in section II.4.2, the finite-state management has been chosen to describe the curse of a dialogue. Indeed, it is actually more suitable for visual

representation and easily understandable. Of course, it has several drawbacks since it doesn't allow the design of all kinds of dialogues but it is the better choice for easy design by novices. Moreover, mapping state-transition networks to an XML script structure is easier. Finally, as said in section III.2.1, an MDP can be described as a state-transition network.

Several methods to use the learning algorithm's results can be investigated. Of course, a display of the state-transition network representing the complete optimal strategy learned by the RL system could be drawn. This is only possible when there are few states in the network and this possibility has been implemented. Nevertheless, when the state space is large and the number of transitions grows, for example in the case of a mixed-initiative dialogue (because the user can be over-informative and provide more information than asked by the system), considering all the possible paths will result in an unmanageable display. This is augmented if repair mechanisms are included in the dialogue.

To overcome this problem, an aided design has been implemented, in addition. Indeed, as the learned strategy is a mapping between states and actions, the designer can be advised to perform a given action when he/she draws a new state according to the Q-value of each value in this state. Actually, several actions can be proposed in decreasing order according to their Q-value.

During the learning process, the learner can also build a model of the environment (exactly like it could do it during online-learning since the learner doesn't take any advantage of the knowledge of the environment) in terms of transition probability and reward distribution estimates. According to the transition probability estimates, the design interface can infer the current state and propose possible successors according to the action chosen. The designer can anyway correct this. After his choice, the designer will have to complete the options of the chosen action (prompts and grammars when the action corresponds to a user query, for example). The graph representation of the dialogue can then be easily drawn with a minimum of prior knowledge.

Once the graph has been drawn, it has to be translated in a set of VoiceXML scripts. This is actually a quite straightforward job. A template is generated for each possible action type. Indeed, constraining questions have all the same structure (a prompt and a recognition process with the loading of a speech grammar). Thus, scripts actually accomplish actions and results of those actions are then passed to the dialogue manager that has been implemented by PHP scripts. The dialogue manager builds the new state and chooses the new action to perform thanks to a look up table filled according to the drawn graph.

There are other possible outputs than VoiceXML scripts. For example, other XML-based languages have been proposed for representation of state-transition networks like the proposed XML Transition Network Definition (XTND) [Nicol, 2000]. On another hand, a new VoiceXML-based multimodal scripting language has been recently proposed: the XHTML + Voice Profile (XHTML+V) [Axelson *et al*, 2004]. This language is a mixture of the usual HTML/XHTML used for website design and of the VoiceXML language. HTML/XHTML already allowed users to provide information thanks to edit boxes, combo boxes, buttons and others. With the XHTML+V each possible input is coupled with a voice-enabled counterpart and the interaction is considered as multimodal since multiple input methods are possible. It is more related to a multichannel interface since information is retrieved either by text or speech inputs but not by providing complementary information by both means. Yet it could be very interesting to produce visual information in order to enhance the interaction but the optimisation process should then take this into account.

X.2. Data Collection

It is an obviousness that the work presented in this text suffers from the lack of online testing and data collection for the training of models. Release of an SDS generated by the described method is the only way to evaluate and validate the accuracy of the simulation and learning methods developed in this text. Yet, all previous researches in the field were so far generally based on an existing dialogue system, which has to be enhanced [Walker *et al*, 1998 a] and from which corpora could be more easily retrieved. It made the methods and results very task-dependent. Attempts to use existing public dialogue corpora demonstrated that they were not suitable for this kind of study since they were often user-initiated [Levin *et al*, 2000].

The design of a prototype and the release to users to get a corpus is time-consuming and could not be done as a preliminary work in the framework of this thesis. Nevertheless, it is in the aim of creating such a prototype that the software described in the previous section has been developed.

X.3. Online Learning

The RL algorithms described in the third part of this text have been initially developed for online learning. Nevertheless, the algorithm necessitates lots of dialogue sessions to converge, which is not suitable for learning with real user. Yet, it is possible to continue the learning process when the prototype has been realised since off-policy learning is a particular feature of the Q-learning algorithm and it is possible to learn optimal policy online while following the previously learned policy. Particularly, since a pure probabilistic model (transition probabilities and reward distribution) has been built in order to create the graphical representation described in the above, it can also be used as a starting point of a Dyna-Q learning algorithm [Sutton & Barto, 1998], which could lead to significant improvement of the strategy.

On another hand, online learning of the user model (for simulation but also for world knowledge: goal inferring, grounding, user adaptation…) can be investigated. Bayesian approach to user modelling is suitable for online learning. Going a step further, learning accurate user model parameters can be considered as a sub-goal of the system. Thus, incorporating this sub-goal in the RL algorithm could lead to other strategies making the system to perform sub-optimal actions in term of short-term dialogue performance to discover user model parameters and improve subsequent dialogues. The likelihood of the model given the observations can be used as an objective function.

Finally, it has been shown that the MDP underlying the dialogue is quite difficult to design. The choice of state and action variables is a key point of the design. Some proposals have been made to learn the optimal state and action variables. For instance, the work reported in [Sallans, 2002].

Fifth Part

~

APPENDICES

Appendix A: Phonemes and Articulatory Features

	φ	Kind	Location	Articulation	Lips	Voicing
1	i	vowel	palatal	oral	stretched	voiced
2	e	vowel	palatal	oral	stretched	voiced
3	E	vowel	palatal	oral	stretched	voiced
4	a	vowel	palatal	oral	stretched	voiced
5	e~	vowel	palatal	nose	stretched	voiced
6	y	vowel	postpalatal	oral	rounded	voiced
7	2	vowel	postpalatal	oral	rounded	voiced
8	9	vowel	postpalatal	oral	rounded	voiced
9	@	vowel	postpalatal	oral	rounded	voiced
10	9~	vowel	postpalatal	nasal	rounded	voiced
11	u	vowel	velar	oral	rounded	voiced
12	o	vowel	velar	oral	rounded	voiced
13	O	vowel	velar	oral	rounded	voiced
14	o~	vowel	velar	nasal	rounded	voiced
15	a~	vowel	velar	nasal	rounded	voiced
16	w	semivowel	palatal	oral	rounded	voiced
17	H	semivowel	postalveolar	oral	rounded	voiced
18	j	semivowel	postalveolar	oral	stretched	voiced
19	p	consonant	bilabial	plosive	N/A	unvoiced
20	b	consonant	bilabial	plosive	N/A	voiced
21	m	consonant	bilabial	nasal	N/A	voiced
22	f	consonant	labiodental	constrictive	N/A	unvoiced
23	v	consonant	labiodental	constrictive	N/A	voiced
24	t	consonant	alveodental	plosive	N/A	unvoiced
25	d	consonant	alveodental	plosive	N/A	voiced
26	n	consonant	alveodental	nasal	N/A	voiced
27	s	consonant	alveolar	constrictive	N/A	unvoiced
28	z	consonant	alveolar	constrictive	N/A	voiced
29	S	consonant	postalveolar	constrictive	N/A	unvoiced
30	Z	consonant	postalveolar	constrictive	N/A	voiced
31	k	consonant	velar	plosive	N/A	unvoiced
32	g	consonant	velar	plosive	N/A	voiced
33	N	consonant	velar	nasal	N/A	voiced
34	R	consonant	postvelar	liquidly	N/A	voiced
35	l	consonant	lateral	liquidly	N/A	voiced

Table 37: French phonemes (SAMPA transcriptions) and their articulatory features

Aperture	Anterior				Posterior	
	stretched		rounded		rounded	
	oral	nasal	oral	nasal	oral	nasal
1st	i		y		u	
2nd	e		2		o	
3rd	E	e~	9	9~	O	o~
4th	a				A	a~
			@			

Table 38: French vowels and their aperture degrees

References

[Aho & Ullman, 1972] A. Aho, J. Ullman, '*The Theory of Parsing, Translation, and Compiling*.' Prentice-Hall, 1972.

[Allen, 1987] J. Allen, '*Natural Language Understanding*.' Benjamin Cummings, 1987, Second Edition, 1994.

[Allen *et al*, 1987] J. Allen, S. Hunnicutt, D. Klatt, '*From Text to Speech: the MITalk System*.' Cambridge University Press, Cambridge, 1987.

[Aust & Shroer, 1998] H. Aust, O. Schroer, '*An Overview of the Philips Dialog System*.' In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne Conference Resort Lansdowne, Virginia, USA, February 1998.

[Austin, 1962] J. Austin, '*How to Do Things with Words*.' Harvard University Press, Cambridge, MA, 1962.

[Axelson *et al*, 2004] J. Axelsson *et al,* '*XHTML + Voice Profile 1.2*.' http://www.voicexml.org/specs/multimodal/x+v/12/, 3 February 2004

[Baekgaard, 1996] A. Baekgaard. '*Dialogue Management in a Generic Dialogue Systems*.' In TWLT-11: Dialogue Management in Natural Language Systems, Netherlands, 1996, pp. 123–132.

[Bagein *et al*, 2003] M. Bagein, O. Pietquin, C. Ris, G. Wilfart, '*An Architecture for Voice-Enabled Interfaces over Local Wireless Networks*.' In Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI'03), Orlando, Florida, USA, July 2003.

[Bahlmann & Burkhardt, 2001] C. Bahlmann, H. Burkhardt, '*Measuring HMM Similarity with the Bayes Probability of Error and its Application to Online Handwriting Recognition*.' In Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR '01), Seattle, Washington, 2001, pages 406-411.

[Barto *et al*, 1983] A. Barto, R. Sutton, C. Anderson, '*Neuronlike elements that can solve difficult learning control problems*.' In IEEE Transactions on Systems, Man, and Cybernetics, vol. 13, 1983, pp. 835-846.

[Baum, 1972] L. Baum, '*An Inequality and Associated Maximization in Statistical Estimation for Probabilistic Functions of Markov Processes*.' Inequalities 3, 1972, pp. 1-8.

[Bayes, 1763] T. Bayes, '*An Essay Towards Solving a Problem in the Doctrine of Chances*.' In Philosophical Transactions of the Royal Society of London. vol 53, 1763, pp. 370-418.

[Bellman, 1957 a] R. Bellman, '*Dynamic Programming*.' Princeton University Press, Princeton, NJ, 1957.

[Bennacef et al, 1994] S. Bennacef, H. Bonneau-Maynard, J. Gauvain, L. Lamel, W. Minker, '*A Spoken Language System for Information Retrieval*.' In Proceedings of the 3rd International Conference on Spoken Language Processing (ICSLP'94), Yokohama, September 1994, pp. 1271-1274.

[Bengio, 2003] S. Bengio, '*Multimodal Speech Processing Using Asynchronous Hidden Markov Models*.' In Information Fusion, 2003.

[Beringer *et al*, 2002] N. Beringer, U. Kartal, K. Louka, F. Schiel, U. Türk, '*PROMISE - A Procedure for Multimodal Interactive System Evaluation*.' In Proceedings of the Workshop on Multimodal Resources and Multimodal Systems Evaluation, Las Palmas, Gran Canaria, Spain, 2002.

[Bishop, 1995] C. Bishop, '*Neural Networks for Pattern Recognition*.' Cambridge, UK: Cambridge University Press, 1995.

[Bobrow et al, 1977] D. Bobrow, R. Kaplan, M. Kay, D. Norman, H. Thompson, and T. Winograd, '*GUS, a Frame Driven Dialog System*.' Artificial Intelligence, vol 8, 1977, pp. 155-173.

[Boite et al, 2000] R. Boite, H. Bourlard, T. Dutoit, J. Hancq, H. Leich, '*Traitement de la Parole*.', 2nd Edition, 488 pp., Presses Polytechniques Universitaires Romandes, Lausanne, ISBN 2-88074-388-5, 2000.

[Boole, 1854] G. Boole, '*An Investigation of the Laws of Thought*.' Walton, London, 1854. (Reprinted by Dover Books, New York, 1954).

[Bourlard & Morgan, 1994] H. Bourlard and N. Morgan, '*Connectionist Speech Recognition – A Hybrid Approach*.', Kluwer Academic Publishers, ISBN 0-7923-9396-1, 1994.

[Boutilier *et al*, 1999] C. Boutilier, T. Dean, S. Hanks, '*Decision-Theoretic Planning: Structural Assumptions and Computational Leverage*.' In Journal of AI Research (JAIR) vol. 11, 1999, pp.1-94.

[Bouwman & Hulstijn, 1998] G. Bouwman, J. Hulstijn, '*Dialogue Strategy Redesign with Reliability Measures*.' In Proceedings of the 1st International Conference on Language Resources and Evaluation 1998, pp.191-198

[Bouwman *et al*, 1999] G. Bouwman, J. Sturm, L. Boves, '*Incorporating Confidence Measures in the Dutch Train Timetable Information System Developed in the ARISE project*.' In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'99), 1999, Vol 1, pp.493-496.

[Brachman & Levesque, 1985] R. Brachman, H. Levesque, editors, '*Readings in Knowledge Representation*.' Morgan Kaufmann Publishers, 1985.

[Bylander, 1991] T. Bylander, '*Complexity Results for Planning*.' In Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91), 1991, pp. 274--279.

[Carberry, 1990] S. Carberry '*Plan Recognition in Natural Language Dialogue*.' ACL-MIT Press Series in Natural Language Processing. Bradford Books, MIT Press, Cambridge, Massachusetts, 1990.

[Carletta, 1996] J. Carletta, '*Assessing Agreement on Classification Tasks: the Kappa Statistic*.' Computational Linguistics, 1996, 22(2), 249-254.

[Carre *et al*, 1984] R. Carre, R. Descout, M. Eskenazi, J. Mariani, M. Rossi, '*The French Language Database: Defining, Planning, and Recording a Large Database*.' In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'84), 1984, pp. 42.10.1--42.10.4.

[Chen *et al*, 1998] S. Chen, D. Beeferman, R. Rosenfeld, '*Evaluation Metrics for Language Models*.' In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, Virginia, February 1998, pp. 275-280.

[Chomsky, 1956] N. Chomsky, '*Three Models for Description of Languages*.' IRE. Transaction on Information Theory, pp. 113-124, 1956.

[Chomsky, 1965] N. Chomsky, '*Aspect of the Theory of Syntax*.' MIT Press, Cambridge, MA, 1965.

[Clark & Shaefer, 1989] H. Clark, E. Schaefer, '*Contributing to Discourse*.' Cognitive Science, 13, 1989, pp.259-294.

[Clarkson & Robinson, 1998] P. Clarkson, T. Robinson, '*The Applicability of Adaptive Language Modelling for the Broadcast News Task*.' In Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98), Sydney, Australia, 1998, pp. 1699-1702.

[Cohen & Perrault, 1979] P. Cohen, C. Perrault, '*Elements of a Plan-based Theory of Speech Acts*.' In Cognitive Science, Vol. 3, 1979, pp. 117-212.

[Cooper, 1990] G. Cooper, '*The Computational Complexity of Probabilistic Inference using Bayesian Belief Networks*.' In Artificial Intelligenc, vol. 42, 1990, pp. 393-405.

[Crystal, 1969] D. Crystal, '*Prosodic Systems and Intonation in English*.' Cambridge Studies in Linguistics, Cambidge University Press, 1969.

[Daelemans & Van den Bosch, 1993] W. Daelemans, A. Van den Bosch, '*Tabtalk: Reusability in Data-Oriented Grapheme-to-Phoneme Conversion*.' In Proceedings of the 3rd European Conference on Speech Technology (Eurospeech'93), Berlin, 1993, pp. 1459-1466.

[Dahlbäck & Jönsson, 1992] N. Dahlbäck, A. Jönsson, '*An Empirically Based Computationally Tractable Dialogue Model*.' In Proceedings of the 14th Annual Conference of the Cognitive Science Society (COGSCI'92), Bloomington, Indiana, July 1992.

[Dahlbäck *et al*, 1993] N. Dahlbäck, A. Jönsson, L. Ahrenberg, '*Wizard of Oz Studies: Why and How*.' In Proceedings of the International Workshop on Intelligent User Interfaces, Orlando, FL, ACM Press, 1993, pp. 193-200

[Dalrymple *et al*, 1991] M. Dalrymple, S. Shieber, F. Pereira, '*Ellipsis and Higher-Order Unification*.' In Linguistics and Philosophy, 1991, pp. 399-452.

[Danieli & Gerbino, 1995] M. Danieli, E. Gerbino, '*Metrics for Evaluating Dialogue Strategies in a Spoken Language System*.' In Working Notes of the AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation, Stanford, California, March 1995, pages 34-39.

[Davis et al, 52] K. Davis, R. Biddulph, S. Balashek, '*Automatic Recognition of Spoken Digits*.', Journal of Acoustic Society of America, 24(6), 1952, pp. 637-642.

[Davis & Hirschberg, 1988] J. Davis, J. Hirschberg, '*Assigning Intonational Features in Synthesized Spoken Directions*.' In Proceedings of the 26th Annual Meeting of the Association for ComputationalLinguistics, Buffalo, 1988, pp. 187-193.

[Dempster *et al*, 1977] A. Dempster, N. Laird, D. Rubin, '*Maximum Likelihood from Incomplete Data via the EM Algorithm*.' J. Royal Statistical Society Series B 39, 1977, pp. 1-38.

[Denecke, 2000] M. Denecke, '*Informational Characterization of Dialogue States*.' In Proceedings of the International Conference on Speech and Language Processing, Beijing, China, 2000.

[Devillers & Bonneau-Maynard, 1998] L. Devillers & H. Bonneau-Maynard. '*Evaluation of Dialog Strategies for a Tourist Information Retrieval System*.' In Proceedings of the 5th International Conference on Speech and Language Processing (ICSLP'98), Sydney, Australia, vol. 4, Dec 1998, pp. 1187-1190.

[Dudley et al, 1939] H. Dudley, R. Riesz, S. Watkins, '*A Synthetic Speaker*.', Journal of Franklin Institute, Philadelphia, 227, 1939, pp. 739-764.

[Dupont & Luettin, 2000] S. Dupont, J. Luettin '*Audio-Visual Speech Modeling for Continuous Speech Recognition*.' IEEE Transactions on Multimedia, 2000.

[Durston et al, 2001] P. Durston, M. Farrell, D. Attwater, J. Allen, H. Kwang J. Kuo, M. Afify, E. Fosler-Lussier, C.-H. Lee, '*OASIS Natural Language Call Steering Trial*.' In Proceedings of the 7th European Conference on Speech Technology (Eurospeech'01), Aalborg, 2001, pp 1232-1235.

[Dutoit, 1993] T. Dutoit, '*High Quality Text-to-Speech Synthesis of the French Language*.' PhD Dissertation, Faculté Polytechnique de Mons, 1993.

[Dutoit, 1997] T. Dutoit, '*An Introduction to Text-To-Speech Synthesis*.' Kluwer Academic Publishers, Dordrecht, 320 pp., ISBN 0-7923-4498-7, 1997.

[Eckert *et al*, 1997] W. Eckert, E. Levin, R. Pieraccini, '*User Modeling for Spoken Dialogue System Evaluation*.' In Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'97), 1997, pp. 80-87.

[Eckert *et al*, 1998] W. Eckert, E. Levin, R. Pieraccini, '*Automatic Evaluation of Spoken Dialogue Systems*.' Technical Report TR98.9.1, AT&T Labs Research, 1998.

[Edmonds 1993] P. Edmonds, '*A Computational Model of Collaboration on Reference in Direction-Giving Dialogues*.' Master's thesis, Computer Systems Research Institute, Department of Computer Science, University of Toronto, October 1993.

[Fisher, 2001] G. Fischer, *User Modeling in Human-Computer Interaction*.' In User Modeling and User-Adapted Interaction, vol. 11(1&2), 2001, pp.65-86.

[Forgie & Forgie, 1959] J. Forgie and C. Forgie, '*Results Obtained from a Vowel Recognition Computer Program*.' Journal of Acoustic Society of America, 31(11), 1959, pp. 1480-1489.

[Freedman, 2000] R. Freedman, '*Plan-Based Dialogue Management in a Physics Tutor*.' In Proceedings of the 6th Applied Natural Language Processing Conference, Seattle, WA, 2000, pp. 52 - 59.

[Garg et al, 2000] A. Garg, V. Pavlovic, J. Rehg, and T.-S. Huang, '*Audio-Visual Speaker Detection using Dynamic Bayesian Networks*.' In Proceedings of 4th International Conference on Automatic Face and Gesture Recognition, 2000, pp. 374-471.

[Gazdar & Mellish, 1989] G. Gazdar, C. Mellish, '*Natural Language Programming in PROLOG*.', Reading, MA: Addison-Wesley, 1989.

[Ghias et al, 1995] A. Ghias, H. Logan, D. Chainberlin, B. Smith, '*Query by Humming: Musical Information Retrieval in an Audio Database*.' In Proceedings of the ACM Conference on Multimedia, 1995, pp. 231-236.

[Goddeau *et al*, 1996] D. Goddeau, H. Meng, J. Polifroni, S. Sene , and S. Busayapongchai, '*A Form-Based Dialogue Manager for Spoken Language Applications*.' In Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP'96), Philadelphia, PA, 1996, pp. 701-704.

[Goldberg *et al*, 1994] E. Goldberg, N. Driedger, R. Kittredge, '*Using Natural-Language Processing to Produce Weather Forecasts*.' IEEE Expert, 1994, pages 45-53.

[Graesser et al, 2001] A. Graesser, K. VanLehn, C. Rosé, P. Jordan, D. Harter, '*Intelligent Tutoring Systems with Conversational Dialogue*.' AI Magazine vol. 22(4) , 2001, pp. 39-52.

[Grosz, 1974] B. Grosz, '*The Structure of Task Oriented Dialogs*.' In Proceedings of the IEEE Symposium on Speech Recognition. Pittsburgh, PA: Carnegie-Mellon University, 1974.

[Grosz & Sidner, 1986] B. Grosz, C. Sidner, '*Attentions, Intentions and the Structure of Discourse*.' Computational Linguistics, 12, 1986, pp. 175–204.

[Hakulinen et al, 2003] J. Hakulinen, M. Turunen, E.-P. Salonen, '*Agents for Integrated Tutoring in Spoken Dialogue Systems*.' In Proceedings of the 8th European Conference on Speech Technology (Eurospeech'03), Genova, 2003.

[Heckerman, 1995] D. Heckerman, '*A Tutorial on Learning with Bayesian Networks*.' Technical Report MSR-TR-95-06, Microsoft Research, 1995

[Hermansky, 1990] H. Hermansky, '*Perceptual Linear Prediction (PLP) Analysis for Speech*.' Journal of the Acoustical Society of America, vol. 87, April 1990, pp. 1738-1752.

[Hirschman *et al*, 1990] L. Hirschman, D. Dahl, D. McKay, L. Norton, M. Linebarger, '*Beyond Class A: A Proposal for Automatic Evaluation of Discourse*.' In Proceedings of the DARPA Speech and Natural Language Workshop, 1990, pp. 109-113.

[Hone & Baber, 1995] K. Hone, C. Baber, '*Using a Simulation Method to Predict the Transaction Time of Applying Alternative Levels of Constraint to User Utterances within Speech Interactive Dialogues*.' In Proceedings of the ESCA Tutorial and Research Workshop on Spoken Dialogue Systems, Hanstholm, Denmark, 1995, pp. 209-213.

[Horvitz *et al*, 1998] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, K. Rommelse, '*The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users*.' Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, July 1998.

[Houghton & Isard, 1987] G. Houghton, S. Isard '*Why to Speak, What to Say, and How to Say It*.' In P. Morris (ed.), Modelling Cognition, Wiley, 1987, 249-267.

[Howard & Matheson, 1984] R. Howard, J. Matheson, '*Influence Diagrams*.' In Principles and Applications of Decision Analysis, vol. 2, Menlo Park, California: Strategic Decision Group, 1984.

[Hsu, 2002] F.-H. Hsu, , '*Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*.' Princeton University Press, 2002.

[Hunt & Black, 1996] A. Hunt, A. Black, '*Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database*.' In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'96), Atlanta, GA, May 1996, pp. 373-376.

[Jelinek, 1976] F. Jelinek, '*Continuous Speech Recognition by Statisical Methods*.' IEEE Proceedings 64:4, 1976, pp. 532-556.

[Jelinek *et al*, 1977] F. Jelinek, R. Mercer, L. Bahl, J. Baker, '*Perplexity - A Measure of Difficulty of Speech Recognition Tasks*.' 94th Meeting of the Acoustic Society of America, Miami Beach, Florida, December 1977.

[Jelinek, 1990] F. Jelinek, '*Self-Organized Language Modeling for Speech Recognition*.' In A. Waibel and K.-F. Lee, editors, Readings in Speech Recognition, Morgan Kaufmann, 1990, pp. 450-506.

[Jönsson & Dahlbäck, 1988] A. Jönsson, N. Dahlbäck, '*Talking to a Computer is not Like Talking to Your Best Friend*.' In Proceedings of The 1st Scandinivian Conference on Artificial Intelligence, Tromsø, Norway, March 9-11, 1988.

[Jourdan *et al*, 1999] M. Jordan, Z. Ghahramani, T. Jaakkola, L. Saul, '*An Introduction to Variational Methods for Graphical Models*.' Machine Learning, vol. 37(2), 1999, pp. 183-233.

[Juang *et al*, 1986] B. Juang, S. Levinson, M. Sondhi, '*Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains*.' In IEEE Transaction on Information Theory, vol. 32, March 1986, pp. 307-309.

[Kamm *et al*, 1998] C. Kamm, D. Litman, M. Walker, '*From Novice to Expert: The Effect of Tutorials on User Expertise with Spoken Dialogue Systems*.' In

Proceedings of the 6th International Conference on Spoken Language Processing, (ICSLP'98), 1998, pp. 1211-1214.

[Kartunnen, 1976] L. Kartunnen, 'D*iscourse Referents*.' In J. McCawley, editor, Syntax and Semantics 7, Academic Press, 1976, pages 363-385.

[Kass & Finin, 1988] R. Kass, T. Finin, '*Modeling the User in Natural Language Systems*.' In Computational Linguistics, vol. 14(3), September 1988, pp. 5-22.

[Kelley, 1984] J. Kelley, '*An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications*.' ACM Transactions on Office Information Systems, 2(1), 1984, pp. 26-41.

[von Kempelen, 1791] W. von Kempelen, '*Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine*.' Wien, J.V. Degen, 1791.

[Klemmer *et al*, 2000] S. Klemmer, A. Sinha, J. Chen, J. Landay, N. Aboobaker, A. Wang, '*SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces*.' In CHI Letters: Proceedings of the ACM Symposium on User Interface Software and Technology, vol. 2, 2000, pp. 1-10.

[Komatani & Kawahara, 2000] K. Komatani & T. Kawahara, '*Generating Effective Confirmation and Guidance Using Two-Level Confidence Measures for Dialogue Systems*.' In Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP'00), Vol.2, 2000, pp.648-651.

[Lamel *et al*, 1991] L. Lamel, J. Gauvain, M. Eskenazi, '*BREF, a Large Vocabulary Spoken Corpus for French*.' In Proceedings of the 2nd European Conference on Speech Technology (Eurospeech'91), Genova, Italy, 1991, pp. 505-508.

[Larsen, 1999] L. Larsen, '*Combining Objective and Subjective Data in Evaluation of Spoken Dialogues*.', In Proceedings of the ESCA Workshop on Interactive Dialogue in Multi-Modal Systems (IDS'99), Kloster Irsee, Germany, 1999, pp. 89-92.

[Larsen, 2003] L. Larsen, '*Issues in the Evaluation of Spoken Dialogue Systems Using Objective and Subjective Measures*.' In Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'03), St. Thomas, U.S. Virgin Islands, 2003.

[Lau et al, 1997] R. Lau, G. Flammia, C. Pao, V. Zue, '*WebGalaxy – Integrating Spoken Language and Hypertext Navigation*.' In Proceedings of the 5th European Conference on Speech Technology (Eurospeech'97), Rhodes, Greece. 1997, pp. 883-886.

[Lavoie *et al*, 1997] B. Lavoie, O. Rambow, E. Reiter, '*Customizable Descriptions of Object-Oriented Models*.' In Proceedings of the Conference on Applied Natural Language Processing (ANLP'97), Washington, DC, 1997.

[Lecoeuche, 2001] R. Lecoeuche, '*Learning Optimal Dialogue Management Rules by Using Reinforcement Learning and Inductive Logic Programming*.' In Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'01), 2001.

[Levenshtein, 1966] V. Levenshtein, '*Binary Codes Capable of Correcting Deletions, Insertions and Reversals*.' Soviet Physics Daklady, vol. 10, 1966, pp. 707-710.

[Levin & Pieraccini, 1997] E Levin, R Pieraccini, '*A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies*,' In Proceedings of the 5th European Conference on Speech Technology (Eurospeech'97), Rhodes, Greece, 1997, pp. 1883-1886.

[Levin *et al*, 1997] E. Levin, R. Pieraccini, W. Eckert, '*Learning Dialogue Strategies within the Markov Decision Process Framework*.' In Proceedings of the IEEE

Workshop on Automatic Speech Recognition and Understanding, Santa Barbara, California, 1997.

[Levin *et al*, 1998] E. Levin, R. Pieraccini, W. Eckert, '*Using Markov Decision Process for Learning Dialogue Strategies*.' In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'98), vol. 1, Seattle, U.S., May 1998, pp. 201-204.

[Levin *et al*, 2000] E. Levin, R. Pieraccini, W. Eckert, '*A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies*.' In IEEE Transactions on Speech and Audio Processing, Vol. 8, No. 1, January 2000, pp. 11-23.

[Lewin, 2000] I. Lewin, '*A Formal Model of Conversational Games Theory*.' In Proceedings of the 4th Workshop on the Semantics and Pragmatics of Dialogues, GOTALOG'00, Gothenburg, 2000.

[Lin & Lee, 2001] B.-S. Lin, L.-S. Lee, '*Computer-Aided Analysis and Design for Spoken Dialogue Systems Based on Quantitative Simulations*.' In IEEE Transactions on Speech and Audio Processing, Vol.9, No.5, July 2001, pp. 534-548.

[Litman *et al*, 1999] D. Litman, M. Walker, M. Kearns, '*Automatic Detection of Poor Speech Recognition at the Dialogue Level*.' In Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics , ACL99, 1999, pp. 309-316.

[Litman & Forbes, 2003] D. Litman, K. Forbes, '*Recognizing Emotions from Student Speech in Tutoring Dialogues*.' In Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'03), St. Thomas, Virgin Islands, November-December, 2003.

[López-Cózar *et al*, 2003] R. López-Cózar, A. de la Torre, J. Segura, A. Rubio '*Assesment of Dialogue Systems by Means of a New Simulation Technique*.' Speech Communication, vol. 40, 2003, pp. 387-407.

[Louloudis *et al*, 2001] D. Louloudis, K. Georgila, A. Tsopanoglou, N. Fakotakis, G. Kokkinakis, '*Efficient Strategy and Language Modeling in Human-Machine Dialogues*.' In Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'01), Vol. XIII, Orlando, Florida, USA, 2001, pp. 229-234.

[Lyngsø *et al*, 1999] R. Lyngsø, C. Pedersen, H. Nielsen, '*Metrics and Similarity Measures for Hidden Markov Models*.' In Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB'99) , 1999, pp. 178-186.

[McGlashan *et al*, 2004] S. McGlashan et al, W3C '*Voice Extensible Markup Language (VoiceXML) Version 2.0*', W3C Proposed Recommendation 3, http://www.w3.org/TR/2003/CR-voicexml20-20030128/, February 2004.

[Mackay, 1999] D. Mackay, '*Introduction to Monte Carlo Methods*.' In M. Jordan, editor, Learning in Graphical Models. MIT Press, 1999.

[McInnes *et al*, 1999] F. McInnes, I. Nairn, D. Attwater, M. Edgington, M. Jack, '*A Comparison of Confirmation Strategies for Fluent Telephone Dialogues*.' In Proceedings of the 17th International Symposium on Human Factors in Telecommunication (HFT'99), 1999, pp. 81-89.

[McRoy *et al*, 2000] S. McRoy, S. Channarukul, S. Ali, '*Text Realization for Dialog*.' In Proceedings of the International Conference on Intelligent Technologies, Bangkok, Thailand, 2000.

[McTear, 1998] M. McTear, '*Modelling Spoken Dialogues with State Transition Diagrams: Experiences with the CSLU Toolkit*.' In Proceedings of the 5th

International Conference on Spoken Language Processing (ICSLP'98), Sydney, Australia, 1998.

[Matsusaka et al, 2001] Y. Matsusaka, T. Tojo, T. Kobayashi, '*Conversation Robot Participating in Group Conversation*.' Journal of IEICE, D-II, vol. J84-D-II, No.6, 2001, pp.898-908.

[Meng *et al*, 1999] H. Meng, W. Lam, K. Low, '*Learning Belief Networks for Language Understanding*.' In Proceedings of the International Workshop on Automatic Speech Recognition and Understanding (ASRU'99), 1999

[Meng *et al*, 2000] H. Meng, C. Wai, R. Pierracinni, '*The Use of Belief Networks for Mixed-Initiative Dialog Modeling*.' In Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP'00), Beijing, China, Oct. 2000.

[Meng *et al*, 2001] H. Meng, et al., '*ISIS: A Trilingual Conversational System with Learning Capabilities and Combined Interaction and Delegation Dialogs*.' In Proceedings of the National Conference on Man-Machine Speech Communication (NCMMSC6), Shenzhen, November 2001.

[Mengusoglu & Ris, 2001] E. Mengusoglu, C. Ris, '*Use of Acoustic Prior Information for Confidence Measure in ASR Applications*.' In Proceedings of the 7th European Conference on Speech Processing, (Eurospeech'01) Scandinavia, Aalborg, September 2001.

[Minsky, 1954] M. Minsky, '*Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem*.' PhD thesis, Princeton University, 1954.

[Minsky, 1975] M. Minsky, '*A Framework for Representing Knowledge*.' In the Psychology of Computer Vision, P. Winston, Ed. New York: McGraw-Hill, 1975, pp. 211--277.

[Montague, 1974] R. Montague, '*Formal Philosophy*.' Yale University Press, New Haven, 1974.

[Murphy, 2001] K. Murphy, '*An Introduction to Graphical Models*.' Informal Notes, May 2001.

[Myers & Rabiner, 1981] C. Myers and L. Rabiner, '*A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected Word Recognition*.' In the Bell System Technical Journal, 60(7), September 1981, pp. 1389-1409.

[von Neumann & Morgenstern, 1944] J. von Neumann, O. Morgenstern, '*Theory of Games and Economic Behavior*.' Princeton University Press, Princeton, New Jersey, 1944.

[Nicol, 2000] G. Nicol '*XTND - XML Transition Network Definition*.' W3C Note 21, http://www.w3.org/TR/2000/NOTE-xtnd-20001121/, November 2000.

[Niimi & Nishimoto, 1999] Y. Niim, T. Nishimoto, '*Mathematical Analysis of Dialogue Control Strategies*.' In Proceedings of the 6th European Conference on Speech Technology, (EuroSpeech'99), Budapest, September 1999.

[D'Orta *et al*, 1987] P. D'Orta, M. Ferretti, S. Scarci, '*Phoneme Classification for Real Time Speech Recognition of Italian*.' In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'87), Dallas, Texas, USA, 1987, pp. 81-84.

[den Os et al, 1999] E. den Os, L. Boves, L. Lamel, P. Baggia, '*Overview of the ARISE Project*.' In Proceedings of the 6th European Conference on Speech Technology, (EuroSpeech'99), Budapest, September 1999, pp. 1527-1530.

[Paek & Horvitz, 2000] T. Paek, E. Horvitz '*Grounding Criterion: Toward a Formal Theory of Grounding*.' Microsoft Research Technical Report, MSR-TR-2000-40, April 2000.

[Pearl 1988] J. Pearl, '*Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*.' Morgan Kaufmann Publishers, Inc. San Francisco, California, 1988.

[Peng & Williams, 1994] J. Peng and R. Williams, '*Incremental Multi-Step Q-Learning*.' In Proceedings of the 11th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 1994, pp. 226-232.

[Pieraccini & Levin, 1992] R. Pieraccini, E. Levin, '*Stochastic Representation of Semantic Structure for Speech Understanding*,' Speech Communication, vol. 11, 1992, pp. 238-288.

[Pietquin & Dutoit, 2002] O. Pietquin, T. Dutoit, '*Modélisation d'un Système de Reconnaissance dans le Cadre de l'Evaluation et l'Optimisation Automatique des Systèmes de Dialogue*.' In Proceedings of the 'Journées d'Etude de la Parole' (JEP'02), Nancy, France, June 2002.

[Pietquin & Dutoit, 2003] O. Pietquin, T. Dutoit, '*Aided Design of Finite-State Dialogue Management Systems*.', In Proceeding of the IEEE International Conference on Multimedia & Expo (ICME'03), Baltimore, july 2003.

[Pietquin & Renals, 2002] O. Pietquin, S. Renals, '*ASR System Modeling for Automatic Evaluation and Optimization of Dialogue Systems*.' In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP'02), Orlando, May 2002.

[Polifroni *et al*, 1992] J. Polifroni, L. Hirschman, S. Seneff, V. Zue, '*Experiments in Evaluating Interactive Spoken Language Systems*.' In Proceedings of the DARPA Speech and Natural Language Workshop, Harriman, NY, February 1992, pp. 28--33.

[Potjer *et al*, 1996] J. Potjer, A. Russel, L. Boves, E. den Os, '*Subjective and Objective Evaluation of Two Types of Dialogues in a Call Assistance Service*.' In IEEE Third Workshop: Interactive Voice Technology for Telecommunications Applications, IVTTA, 1996, pp. 89-92.

[Power, 1979] R. Power, '*The Organization of Purposeful Dialogues*.' Linguistics 17, 1979, pp. 107-152.

[Printz & Olsen, 2000] H. Printz, P. Olsen, '*Theory and Practice of Acoustic Confusability*.' In Proceedings of ISCA ITRW ASR2000 Workshop, Paris, France, 2000, pp. 77-84.

[Quek et al, 2002] F. Quek, D. McNeill, B. Bryll, S. Duncan, X. Ma, C. Kirbas, K. McCullough, R. Ansari, '*Multimodal Human Discourse: Gesture and Speech*.' ACM Transactions on Computer-Human Interaction (ToCHI). Vol. 9, No. 3, September 2002, Pages 1-23.

[Rabiner & Shafer, 1978] L. Rabiner, R. Schafer, '*Digital Processing of Speech Signals*.' Englewood Cliffs, New Jersey: Prentice-Hall, 1978.

[Rabiner & Juang, 1993] L. Rabiner, B.H. Juang, '*Fundamentals of Speech Recognition*.' Prentice Hall, Signal Processing Series, 1993.

[Rahim *et al*, 2001] M. Rahim, G. Di Fabbrizio, C. Kamm, M. Walker, A. Pokrovsky, P. Ruscitti, E. Levin, S. Lee, A. Syrdal, K. Schlosser, '*Voice-IF: a Mixed-Initiative Spoken Dialogue System for AT&T Conference Services*.' In Proceedings of the 7th European Conference on Speech Processing, (Eurospeech'01), Scandinavia, Aalborg, 2001.

[Rambow *et al*, 2001] O. Rambow, S. Bangalore, M. Walker, '*Natural Language Generation in Dialog Systems*.' In Proceedings of the 1st International Conference on Human Language Technology Research (HLT'01), San Diego, USA, 2001.

[Reichman, 1981] R. Reichman, '*Plain-Speaking: A Theory and Grammar of Spontaneous Discourse*.' Ph.D. Thesis, Department of Computer Science, Harvard University, Cambridge, Massachusetts, 1981.

[Reiter & Dale, 2000] E. Reiter, R. Dale, '*Building Natural Language Generation Systems*.' Cambridge University Press, Cambridge, 2000.

[Rich, 1979] E. Rich, '*User Modeling via Stereotypes*.' In Cognitive Science, vol. 3(4), 1979, pp. 329-354.

[Ristad & Yianilos, 1998] E. Ristad, P. Yianilos, '*Learning String-Edit Distance*.' In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20 (5), 1998, pp. 522-532.

[Rose et al, 1999] C. Rose, B. Di Eugenio, J. Moore, '*A Dialogue Based Tutoring System for Basic Electricity and Electronics*.' In Proceedings of AI in Education, 1999.

[Roy *et al*, 2000] N. Roy, J. Pineau, S. Thrun, '*Spoken Dialogue Management Using Probabilistic Reasoning*.' In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000) , Hong Kong, 2000.

[Sallans, 2002] B. Sallans, '*Reinforcement Learning for Factored Markov Decision Processes*.' Ph.D. Thesis, Department of Computer Science, University of Toronto, 2002.

[Sawhney & Schmandt, 1998] N. Sawhney, C. Schmandt, '*Speaking and Listening on the Run: Design for Wearable Audio Computing*.' In IEEE International Symposium on Wearable Computing, 1998.

[Schwartz, 1978] G. Schwartz, '*Estimating the Dimension of a Model*.' In Annals of Statistics, vol. 6, 1978, pp. 497-511.

[Science News Letter, 1939] '*Now a Machine That Talks with the Voice of Men*.' Science News Letter, January 14, 1939, page 19.

[Shannon, 1948] C. Shannon, '*A Mathematical Theory of Communication*.' Bell System Technical Journal, vol. 27, 1948, pp. 379-423, 623-656.

[Shannon, 1950 a] C. Shannon, '*Automatic Chess Player*.' Scientific American 182(2), 1950, pp. 48-51.

[Shannon, 1950 b] C. Shannon, '*Programming a Computer for Playing Chess*.' Philosophical Magazine Vol. 41, 1950, pp. 256-275.

[Scheffler & Young, 1999] K. Scheffler, S. Young, '*Simulation of Human-Machine Dialogues*.' Technical Report CUED/F-INFENG/TR 355, Cambridge University Engineering Dept, 1999.

[Scheffler & Young, 2002] K. Scheffler, S. Young, '*Automatic Learning of Dialogue Strategy using Dialogue Simulation and Reinforcement Learning*.' In Proceeding of Human Language Technology 2002, San Diego, pp. 12-18.

[Shieber *et al*, 1990] S. Shieber, G. van Noord, F. Pereira, R. Moore, '*Semantic Head-Driven Generation*.' Computational Linguistics, 16:30-42, 1990.

[Sidner, 1983] C. Sidner, '*Focusing in the Comprehension of Definite Anaphora*.' In Computational Models of Discourse, M. Brody and R. Berwick, eds. Cambridge, Mass: MIT Press, 1983, pp. 267-330.

[Simpson & Fraser, 1993] A. Simpson, N. Fraser, '*Black Box and Glass Box Evaluation of the SUNDIAL System*.' In Proceedings of the 3rd European Conference on Speech Communication and Technology (Eurospeech'93), Berlin, Germany, 1993, pp. 1423-1426.

[Singh *et al*, 1999] S. Singh, M. Kearns, D. Litman, M. Walker, '*Reinforcement Learning for Spoken Dialogue Systems*.' In Proceedings of the 15th Conference on Neural Information Processing Systems (NIPS'99), Denver, USA, 1999.

[Smith *et al*, 1992] R. Smith, D. Hipp, A. Biermann, '*A Dialog Control Algorithm and its Performance*.' In Proceedings of the 3rd Applied Natural Language Processing, Trento, Italy, 1992, pp. 9-16.

[Smith & Hipp, 1994] R. Smith, R. Hipp, '*Spoken Natural Language Dialog Systems: a Practical Approach*.' Oxford University Press, New York, 1994.

[Sneele & Waals, 2003] P. Sneele, J. Waals, '*Evaluation of a Speech-Driven Telephone Information Service using the PARADISE Framework: a Closer Look at Subjective Measures*.' In Proceedings of the 8th European Conference on Speech Technology, (EuroSpeech'03), Geneva, 2003, pp. 1949-1952.

[Sondik, 1971] E. Sondik, '*The Optimal Control of Partially Observable Markov Processes*.' PhD Thesis, Department of Engineering-Economics Systems, Stanford University, 1971.

[Suryadi & Gmytrasiewicz, 1999] D. Suryadi, P. Gmytrasiewicz, '*Learning Models of Other Agents using Influence Diagrams*.' In Proceedings of the 1999 International Conference on User Modeling, Banf, CA, July 1999, pp. 223-232.

[Sutton, 1988] R. Sutton, '*Learning to Predict by Methods of Temporal Differences*.' In Machine Learning 3, Kluwer Academic Publishers, Boston, 1988, pp. 9-44.

[Sutton & Barto, 1998] R. Sutton, A. Barto, '*Reinforcement Learning: An Introduction*.' Cambridge, MA: MIT Press, 1998.

[Tesauro, 1994] G. Tesauro, '*TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play*.' In Neural Computation, 6(2), 1994, pp. 215-219.

[Turing, 1937] A. Turing, '*On Computable Numbers with an Application to the Entscheidungsproblem*.' In Proceedings of the London Mathematical Society, vol. 42, 1937, pp. 230-265.

[Turing, 1950] A. Turing, '*Computing Machinery and Intelligence*.' Mind 49: 1950, pp. 433-460.

[Tversky, 1977] A. Tversky, '*Features of Similarity*.' In Psychological Review, vol. 84, no. 4, 1977, pp. 327-352.

[Veldhuijzen van Zanten, 1999] Gert Veldhuijzen van Zanten, '*User Modelling in Adaptive Dialogue Management*.' In Proceedings of the 6th European Conference on Speech Communication and Technology, (Eurospeech'99), Budapest, Sept. 1999, volume 3, pp. 1183–1186.

[Viterbi, 1967] A. Viterbi, '*Error Bounds for Convolutions Codes and an Asymptotically Optimum Decoding Algorithm*.' In IEEE Transactions on Information Theory, IT-13(2), 1967, pp. 260-269.

[Wachsmuth & Sagerer, 2002] S. Wachsmuth, G. Sagerer, '*Bayesian Networks for Speech and Image Integration*.' In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02), Edmonton, Alberta, Canada, 2002, pp. 300-306.

[Wai *et al*, 2001] C. Wai, H. Meng, R. Pierracinni, '*Scalability and Portability of a Belief Network-based Dialog Model for Different Application Domains*.' In Proceedings of the Human Language Technology Conference, San Diego, March 2001.

[Walker, 1993] M. Walker, '*Informational Redundancy and Resource Bounds in Dialogue*.' Ph.D. Thesis, University of Pennsylvania, 1993.

[Walker *et al*, 1996] M. Walker, J. Cahn, S. Whittaker, '*Linguistic Style Improvisation for Lifelike Computer Characters*.' In Proceedings of the AAAI Workshop AI, Alife and Entertainment, Portland, 1996.

[Walker *et al*, 1997a] M. Walker, D. Litman, C. Kamm, A. Abella, '*PARADISE: A Framework for Evaluating Spoken Dialogue Agents*.' In Proceedings of the

35th Annual Meeting of the Association for Computational Linguistics, Madrid, Spain, 1997, pp. 271-280.

[Walker *et al*, 1997b] M. Walker, D. Hindle, J. Fromer, G. Di Fabbrizio, C. Mestel , '*Evaluating Competing Agent Strategies for a Voice Email Agent*.' In Proceedings of the 5th European Conference on Speech Communication and Technology, (Eurospeech'97), Rhodes, Greece, 1997.

[Walker *et al*, 1998 a] M. Walker, D. Litman, C. Kamm, A. Abella, '*Evaluating Spoken Dialogue Agents with {PARADISE}: Two Case Studies*.' In Computer Speech and Language, 12-3, 1998.

[Walker *et al*, 1998 b] M. Walker, A. Joshi, E. Prince, editors '*Centering Theory in Discourse*.' Oxford University Press, 1998.

[Walker *et al*, 1999] M. Walker, J. Boland, C. Kamm, '*The Utility of Elapsed Time as a Usability Metric for Spoken Dialogue Systems*.' In Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'99), Keystone, Colorado, U.S.A, 1999.

[Walker, 2000] M. Walker, '*An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email*.' In the Journal of Artificial Intelligence Research, JAIR, Vol 12. , 2000, pp. 387-416.

[Walker *et al*, 2000] M. Walker, L. Hirschman, J. Aberdeen, '*Evaluation For Darpa Communicator Spoken Dialogue Systems*.' In Proceedings of the Language Resources and Evaluation Conference, LREC , 2000.

[Walker *et al*, 2001] M. Walker, J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, A. Le, S. Lee, S. Narayanan, K. Papineni, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, G. Sanders, S. Seneff, D. Stallard, S. Whittaker, '*DARPA Communicator Dialog Travel Planning Systems: The June 2000 Data Collection*.' In Proceedings of the 7th European Conference on Speech Technology, (Eurospeech'01), Aalborg, 2001.

[Walker *et al*, 2002] M. Walker, O. Rambow, M. Rogati, '*Training a Sentence Planner for Spoken Dialogue Using Boosting*.' Computer Speech and Language Special Issue on Spoken Language Generation , July 2002.

[Watkins, 1989] C. Watkins, '*Learning from Delayed Rewards*.' Ph.D. Thesis, Psychology Department, Cambridge University, Cambridge, England, 1989.

[Weizenbaum, 1966] J. Weizenbaum, '*ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine*.' Communications of the ACM 9(1), 1966, pp. 36-45.

[Williams & Renals 1997] G. Williams, S. Renals, '*Confidence Measures for Hybrid HMM/ANN Speech Recognition*.' In Proceedings of the 5th European Conference on Speech Technology, (Eurospeech'97), Rhodes, 1997, pp. 1955-1958.

[Winograd, 1972] T. Winograd, '*Understanding Natural Language*.' Academic Press, New York, 1972.

[Woods, 1970] W. Woods, '*Transition Network Grammars for Natural Language Analysis*.' Communications of the ACM, 13:591-606, 1970.

[Zhang *et al*, 2001] B. Zhang, Q. Cai, J. Mao, B. Guo, '*Planning and Acting Under Uncertainty: a New Model for Spoken Dialogue Systems*.' In Proceedings of Uncertainty in Artificial Intelligence (UAI'01), Seattle, WA, 2001, pp.572-570.

[Zue et al, 2000] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T.J. Hazen, L. Hetherington, *'JUPITER: A Telephone-Based Conversational Interface for Weather Information*.' In IEEE Transactions on Speech and Audio Processing, 88(1), 2000.

[Zukerman & Albrecht, 2001] I. Zukerman, D. Albrecht, '*Predictive Statistical Models for User Modeling*.' In User Modeling and User-Adapted Interaction, vol. 11(1-2), 2001, pp. 5-18.