An open-source framework for supporting the design and implementation of natural-language spoken dialog systems

Pierrick Milhorat

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0087

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Signal et Image »

présentée et soutenue publiquement par

Pierrick Milhorat

le 17 décembre 2014

An Open-source Framework for Supporting the Design and Implementation of Natural-language Spoken Dialog Systems

Directeur de thèse : **Gérard CHOLLET**
Co-encadrement de la thèse : **Jérôme BOUDY**

Jury	
Mme Kristiina JOKINEN	Rapporteur
Mr Olivier PIETQUIN	Rapporteur
Mme María Inés TORRES	Examinateur
Mr Charles RICH	Examinateur
Mr David SADEK	Examinateur
Mr Stephan SCHLÖGL	Invité
Mr Gérard CHOLLET	Directeur de thèse
Mr Jérôme BOUDY	Co-encadrant de thèse

T H È S E

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

Acknowledgments

Here, I would like to thank my advisors, Gérard Chollet and Jérôme Boudy.

Gérard offered me this unique opportunity. While he guided throughout the whole journey, he also let me pursue my own directions and take responsibility for it. I thank him for that.

Jérôme has been a wise and kind advisor. His desire to help, his support and the valuable discussion we had are good reasons to be thankful.

I have enjoyed the support and company of my colleagues, Daniel Regis Sarmento Caon, Houssemeddine Khemiri, Jirasri Deslis, Jacques Feldmar, Olivier Meule and Stephan Schlögl. They all played a role in this work.

I would also like to thank the jury members for their review of this document as well as their comments at the defense.

Short Abstract

Recently, global tech companies released so-called virtual intelligent personal assistants. Highlighting what was the emerging trend in the interaction with machines, especially on hand-held devices.

This thesis has a bi-directional approach to the domain of spoken dialog systems. On the one hand, parts of the work emphasize on increasing the reliability and the intuitiveness of such interfaces. On the other hand, it also focuses on the design and development side, providing a platform made of independent specialized modules and tools to support the implementation and the test of prototypical spoken dialog systems technologies.

The topics covered by this thesis are centered around an open-source framework for supporting the design and implementation of natural-language spoken dialog systems. The framework has been developed and set up following the use cases of the supporting projects. It is still currently evolving.

One way to characterize a spoken dialog system is by using the listening method it is applying. Continuous listening, where users are not required to signal their intent prior to speak, has been and is still an active research area. Two methods are proposed here, analyzed and compared.

According to the two directions taken in this work, the natural language understanding subsystem of the platform has been thought to be intuitive to use, allowing a natural language interaction. It is easy to set up as well since one does not need much knowledge of the technologies involved to configure the subsystem for one's application.

Finally, on the dialog management side, this thesis argue in favor of the deterministic modeling of dialogs. However, such an approach requires intense human labor, is prone to error and does not ease the maintenance, the update or the modification of the models. A new paradigm, the linked-form filling language, offers to facilitate the design and the maintenance tasks by shifting the modeling to an application specification formalism.

Résumé court

L'interaction vocale avec des systèmes automatiques connaît, depuis quelques années, un accroissement dans l'intérêt que lui porte tant le grand public que la communauté de la recherche. Cette tendance s'est renforcée avec le déploiement des assistants vocaux personnels sur les terminaux portables.

Cette thèse s'inscrit dans ce cadre pour aborder le sujet depuis deux points de vue complémentaires. D'une part, celui apparent de la fiabilité, de l'efficacité et de l'utilisabilité de ces interfaces. D'autre part, les aspects de conception et d'implémentation sont étudiés pour apporter des outils de développement aux concepteurs plus ou moins initiés de tels systèmes.

A partir des outils et des évolutions dans le domaine, une plate-forme modulaire de dialogue vocal a été agrégée. Progressivement, celle-ci a été configurée pour répondre aux exigences des scénarios d'usage et de démonstration dans l'optique des collaborations encadrant ce travail. Le système s'est complexifié et est constamment en évolution suivant les approches mentionnées plus haut.

L'interaction continue, basée sur une "écoute" permanente du système pose des problèmes de segmentation, de débruitage, de capture de son, de sélection des segments adressés au système, etc... Une méthode simple, basée sur la comparaison des résultats de traitements parallèles a prouvé son efficacité, tout comme ses limites pour une interaction continue avec l'utilisateur.

Les modules de compréhension du langage forment un sous-système interconnecté au sein de la plate-forme. Ils sont les adaptations d'algorithmes de l'état de l'art comme des idées originales. Ils ont été pensé pour rendre l'interaction naturelle et fiable tout en limitant la complexité de leur configuration et en maintenant leur généricité et donc leur usage à travers plusieurs dialogues. L'utilisabilité est évaluée à partir de données collectées lors d'essais en laboratoire avec des utilisateurs réels. L'aisance dans la configuration d'un tel système et sa modularité, plus difficiles à prouver empiriquement, sont discutées.

Le choix de la gestion du dialogue basé sur des modèles de tâches hiérarchiques, comme c'est la cas pour la plate-forme, est argumenté. Ce formalisme est basé sur une construction humaine et présente, de fait, des obstacles pour concevoir, implémenter, maintenir et faire évoluer les modèles. Pour parer à ceux-ci, un nouveau formalisme est proposé qui se transforme en hiérarchie de tâches grâce aux outils associés. Ce document se veut être une référence du nouveau langage code et de sa conversion, il présente également des mesures d'évaluation de l'apport d'un tel outil.

Supporting Projects

CompanionAble

Adapted from the CompanionAble's website homepage (http://www.companionable.net)

There are widely acknowledged imperatives for helping the elderly live at home (semi)-independently for as long as possible. Without cognitive stimulation support the elderly dementia and depression sufferers can deteriorate rapidly and the carers will face a more demanding task. Both groups are increasingly at the risk of social exclusion.

CompanionAble's objective is to provide the synergy of Robotics and Ambient Intelligence technologies and their semantic integration to provide for a care-giver's assistive environment. This supports the cognitive stimulation and therapy management of the care-recipient. This is mediated by a robotic companion (mobile facilitation) working collaboratively with a smart home environment (stationary facilitation).

The distinguishing advantages of the CompanionAble Framework Architecture arise from the objective of graceful, scalable and cost-effective integration. Thus CompanionAble addresses the issues of social inclusion and homecare of persons suffering from chronic cognitive disabilities prevalent among the increasing European older population. A participative and inclusive co-design and scenario validation approach drives the Research and Technological Development (RTD) efforts in CompanionAble; involving care recipients and their close carers as well as the wider stakeholders. This is to ensure end-to-end systemic viability, flexibility, modularity and affordability as well as a focus on overall care support governance and integration with quality of experience issues such as dignity-privacy-security preserving responsibilities fully considered.

CompanionAble has been evaluated at a number of testbeds representing a diverse European user-base as the proving ground for its socio-technical-ethical validation. The collaboration of leading gerontologists, specialist elderly care institutions, industrial and academic RTD partners, including a strong cognitive robotics and smart-house capability makes for an excellent confluence of expertise for this innovative project.

Figure 1: CompanionAble logo

Start date: January 2008
Duration: 48 months

Arhome

Arhome is a French national project, which aims at building an unified exchange platform for home care and services. The consortium consists of partners from home care agencies, software development companies and research labs.

The central system of the project's architecture is a communication bus. It provides an access to shared databases made available for senior beneficiaries, authorized relatives and caregivers, according to the access rights they are granted with.

The project is motivated by the scarcity of communication tools in the domain between the seniors and their relatives, the seniors and their caregivers and the relatives and the caregivers.

Interface devices are smartphones for the relatives, tablets for the main users, i.e. the seniors, and a web interface for the caregivers who may also use the seniors' tablet at home. In addition to the Graphical User Interface (GUI), which is available on all devices, the project aims at providing a vocal assistant to guide the senior while he/she is using the system on the tablet.

Figure 2: Arhome logo

Start date: October 2011
Duration: 24 months

vAssist

Adapted from the vAssist website's homepage (`http://vassist.cure.at`)

vAssist stands for Voice Controlled Assistive Care and Communication Services for the Home.

The goal of the vAssist project is to provide specific voice controlled Home Care and Communication Services for two target groups of older persons: seniors suffering from chronic diseases and persons suffering from (fine) motor skills impairments. The main goal is the development of simplified and adapted interface variants for tele-medical and communication applications using multilingual natural speech and voice interaction (and supportive GUIs where necessary).

vAssist aims to enhance the perceived quality of healthcare services and to enable a reduction in the costs related to their production and delivery by achieving channel independence in the delivery of vAssist services, so that existing hardware and interfaces in the home of the users can be used such as PC, TV, mobile phone or tablets. Further, the vAssist consortium considers user, technical and economic constraints in a sound methodological setup throughout the whole project duration (from user requirements to field evaluation studies). From an interface point of view vAssist leverages approaches to connect to universal interfaces in the delivery of Ambient Assisted Living (AAL) services (e.g. UniversAAL, I2Home, etc.) and provides user-specific voice assisted interfaces in order to address a wider audience.

However, vAssist's aim is not to develop another platform for service and interface integration, but to develop specific modules in order to enhance existing services with voice and speech intelligence. Existing platforms like the ones above-mentioned are considered in the exploitation strategy and technical design of vAssist services.

A User-Centered Market-Oriented Design (UCMOD) process involves end-users in all phases of the development process and considers market-oriented aspects from the initial phase of the project. This assures that the iteratively developed services and business model (service and hardware delivery) are adapted to the requirements and needs of the users and show a high market potential within the next 2-3 years. Overall, vAssist services can be ready for market in this time after completion of the project – as it bases on existing services and such that are currently already in use, with service enhancement by adding new interaction without the requirement of new service development from scratch.

Figure 3: vAssist logo

Start date: December 2011
Duration: 36 months

Table of contents

List of Figures

List of Tables

List of Acronyms

AAL	Ambient Assisted Living
AM	Acoustic Model
API	Application Programming Interface
ASR	Automatic Speech Recognition
CC	Context Catcher
CFG	Context-Free Grammar
CGN	Corpus Gesproken Nederlands
CMT	Coincidence Microphone Technology
CMU	Carnegie Mellon University
DA	Dialog Act
DAM	Dialog Act Mapper
DM	Dialog Manager
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HMM	Hidden Markov Model
HTK	Hidden Markov model ToolKit
HTTP	Hypertext Transfer Protocol
HVS	Hidden Vector State
ID	Identifier

IS Information State

KS Knowledge Source

LFF Linked Form-Filling

LM Language Model

LVCSR Large Vocabulary Continuous Speech Recognition

MAP Maximum A Posteriori

MDP Markov Decision Process

MFCC Mel-Frequency Cepstral Coefficient

MLLR Maximum Likelihood Linear Regression

NL Natural Language

NLG Natural Language Generator

NLU Natural Language Understanding

PCFG Probabilistic Context-Free Grammar

POMDP Partially Observable Markov Decision Process

RTD Research and Technological Development

SASSI Subjective Assessment of Speech System Interfaces

SCTK NIST Scoring Toolkit

SDS Spoken Dialog System

SEQ Single Ease Questionnaire

SF Semantic Frame

SOAP Simple Object Access Protocol

SP Semantic Parser

SSA Speech/Sound Analysis

SURR Semantic Unifier and Reference Resolver

SUS System Usability Scale

TTS Text-To-Speech

UCMOD User-Centered Market-Oriented Design

vAssist Voice Controlled Assistive Care and Communication Services for the Home

VoIP	Voice over IP
WoZ	Wizard of Oz
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Abstract

Most recent deployment of global companies' virtual personal assistants such as Siri (Apple), Google Now (Google), Cortana (Microsoft), S Voice (Samsung) or Voice Mate (LG) demonstrated the potential of vocal interaction with automated systems. A Spoken Dialog System (SDS) constitutes such an interface [35]. It is characterized as a system that is able to recognize speech [5, 8, 90, 152, 194, 195], understand the language [7, 48], make decisions based on that [29, 37] and generate a response back. The assistants mentioned earlier are instances of SDSs. The recent interest and success of them is closely related to the increasing quality of Automatic Speech Recognition (ASR) technologies.

This thesis tries to build on this advances, identifying weaknesses and providing solutions. The approach is two-fold.

On the one hand, it recognizes the lack of open-source SDSs for research purposes which hinders the development of systems and their individual components. Consequently, a framework for building SDS has been provided along with tools to support the design and the set up of modules for ASR, Natural Language Understanding (NLU), dialog management, Natural Language (NL) generation and speech synthesis.

On the other hand, studies have defined the compulsory features required to build an SDS. They are reliability, spontaneity, displayed intelligence, human-like behaviour, etc. Each of them has been explored and methods to improve them have been experimented.

Mainly, this thesis makes five proposals to contribute to the SDS domain. They are detailed in the following.

A modular open-source platform for spoken dialog systems An SDS is defined as a system providing an interface to a service, may it be virtual or concrete, via a human-like dialog [35].

Those systems not usually consist of a single component but comprise several specialized programs combined in order to recognize the speech, extract the information relevant to the dialog in the transcriptions, act on back-end services, decide on the best next step, generate NL from Dialog Acts (DAs) and synthesize speech.

While commercial deployed systems are black boxes for obvious monetization considerations, there are open-source systems such as JUPITER [223], the

Philips Automatic Train Timetable Information System [6], Olympus [17] (Let's Go [57, 157, 158]), etc. However, the second group, while freely available, are not so much user-friendly for who would like to build an SDS for one's own service.

This thesis claims to have conceived a modular open-source framework for SDS which is easy to use, easy to set up and within which components can be easily substituted.

Moreover, the platform has been populated with components to fulfill all the requirements drawn from the study of previous SDSs. The following work bases all its implementation within this continuously evolving framework of components.

A Step Towards continuous listening for spoken interaction The pace of the interaction of deployed commercial SDSs is dictated by a binary modality, e.g. a keyword detection mechanism, a motion sensor, a button, etc. Those mechanisms are implemented to indicate the beginning of a user's speech segment, i.e. the user signals his/her intention to address the system before actually producing the utterance.

The turn taking (who is speaking) in human-human dialogs is much more complex. However, such a constraint allows the system to avoid some hurdles in the processing.

Indeed, continuous listening, i.e. when the microphone is kept open throughout the whole dialog and the task of segmenting and filtering the incoming signal is left entirely to the automatic system, requires a few issues to be tackled: noisy signal, speech-free segments, start and end of utterance marking, distant speech recognition, echo removal, out-of-scope utterances, etc.

According to the conditions of deployment, a method, based on sound classification, parallel processing of speech, alignment scoring and attention level computation, has been implemented and evaluated. It tries to deal with some of those issues at once.

A sub-system to map natural-language utterances to situated parametrized dialog acts The NLU community looks at ways to extract knowledge from those modalities from which humans can produce and convert the signal to machine-readable pieces of information. For an SDS, this means that the NLU sub-system converts spoken utterances to representations of the intent of the user commonly known as DAs. Those depend on the current dialog state, and so does the interpretation of the meaning of transcribed segments.

Four compulsory features for a NLU system, integrated to an SDS, have been identified. It has to: allow variations in the utterances of the user, allow for a mixed initiative, integrate the dialog context in the process, integrate relevant external environment variables to augment the information.

The SDS framework developed here includes a modular NLU system which segments the incoming signal, transcribes the speech, parses the utterances, augments the inputs with the dialog context and additional external sources,

selects the user DA and tries to detect and recover from errors. It as been conceived to be convenient to use and human-like while, according to the second axis of this thesis, easy to set up and manipulate.

Linked-form filling: a new paradigm to create and update task-based dialog models In the literature one finds a number of methods to model and control dialogs. They range from the early adjacency-pairs analysis, which scripted the interaction, through the Information State (IS) [101, 134], the flow-graphs, the example databases [91, 106, 108, 109], the Markov Decision Processes (MDPs) [113, 114, 167], the plan-based dialog managers up to the most advanced Partially Observable Markov Decision Processes (POMDPs) [25, 28, 79, 207, 210, 216, 217, 218, 221, 221]. The older approaches require system designers to carry out a thorough analysis of the interaction to be automated and to manually define the domain, the parameters and, to some extent, the behaviour of the machine. MDPs, POMDPs and example-based models are called stochastic models since their strategy for selecting the next most appropriate step(s) is learnt from data.

The context of development along the course of this thesis favoured the employment of a deterministic formalism to control dialogs. This was motivated by the requirements of the overall system it had to be integrated to. Table 1 summarizes the advantages and drawbacks of both approaches with an emphasis on the task-based paradigm for the deterministic part.

Stochastic approach		Deterministic approach	
Advantages	Drawbacks	Advantages	Drawbacks
Only the domain has to be defined	Require prior data collection	Intuitive	Manual modelling of the interaction
Can be automatically adapted	Need target user to be involved in early development	No need for prior data collection	Difficult to build, debug, adapt
Suitable for big data	Exact solution computation may be intractable		Static

Table 1: Comparison of stochastic and deterministic approaches for dialog management

While the design of deterministic dialog models requires the analysis of actual interaction data and the knowledge of an expert, it is generally agreed that the basic workflow of an application is straightforward to build. Thus, the Linked Form-Filling (LFF) proposes to shift the modeling towards a description, in

terms of connected forms, of the service/application to which one wants to add a dialog interface to.

An LFF model consists of a set of forms linking each other under certain conditions and to which grounding scripts can be attached. Such a network is then transformed into an ANSI/CEA-2018 [41] compliant task hierarchy which is suitable for some Dialog Managers (DMs) [18, 20, 72, 112, 161, 162, 163, 164, 165, 182] to handle an interaction.

The LFF layer hides the complexity of plan-based dialog models while preserving alternative paths achieving the same goals.

Chapter 1

Introduction

Lately speech and other types of NL are experiencing an increased acceptance when being used for interacting with "intelligent" computing systems. Companies increasingly provide us with potentially new application scenarios where this modality is seen as the best way of operation. This trend is particularly reflected by recent technology releases such as Apple's Siri, Google's Google Now, Microsoft's Cortana, Samsung's S Voice and LG's Voice Mate.

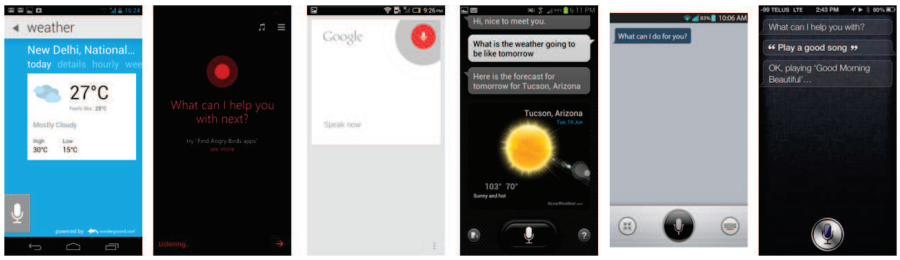


Figure 1.1: Current commercial vocal user interfaces: Maluuba (Maluuba), Cortana (Microsoft), Google Now (Google), S Voice (Samsung), Voice Mate (LG), Siri (Apple)

While these products clearly demonstrate the industry's vision of how we should be interacting with our current and future devices, they also highlight some of the great challenges that still remain. The main criticisms that have been received questions the the reliability, the usefulness, the data protection, the proprietary aspects of the technologies, etc.

In 2010 17.38% of Europe's population was older than 65 years of age and current projections suggest that by 2060 we will have less than two people of working age (15-65 years) for every person beyond 65. Technologies that offer more natural and less cognitive demanding interaction channels, such as speech, may therefore not only attract our technophile young but generally be seen as a way of supporting the life of an ageing society.

However, we often lack the necessary tools and methods that would allow us to realistically develop, test and study these types of interactions. Building, integrating and combining language technologies such as ASR, NLU, DM, Natural Language Generator (NLG) and Text-To-Speech (TTS), for different application scenarios, poses significant design and software engineering challenges [1, 22, 30, 40, 85, 169, 170].

NL interaction is, despite those recent advances, still not reliable enough to be used by a larger range of users and to be accepted as an efficient mean of communication with a machine. We face a socio-technological problem where the use of those error-prone technologies may easily lead to unsatisfying user experiences [86]. While for some talking to a computer may simply convey a great user experience, for others, it can offer a significant alleviation when interacting with a piece of technology. However, the leap forward taken by ASR has demonstrated how a technology entering the virtuous circle of machine learning (Figure 1.2) may benefit from it.

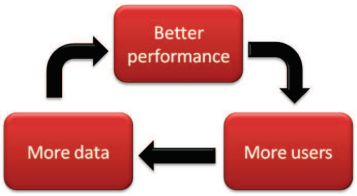


Figure 1.2: Virtuous circle of machine learning

SDSs build on the advances of the technologies that define them. They aim at mimicking the behaviour of a human supporting the spoken interaction. The cognitive abilities that we, humans, can demonstrate is complex in many ways [3]. Thus current systems, even the mosts advanced ones struggle to act like true autonomous-thinking entities, which consequently appear to the user as unnatural [87]. Some challenges to be tackled are the turn taking, the DA extraction from speech, context inclusion, decision making, adaptive learning, or personalization.

Given the status of the technologies related to SDSs, this thesis explores three research questions:

- How can we make the human-machine spoken interaction more reliable?
- What are the important features for an SDS to improve its human-like appearance?
- How can we support the development of SDSs?

1.1 A Modular Open-source Platform for Spoken Dialog Systems

An SDS is a system providing an interface to a service or an application via a dialog. An interaction qualifies as dialog as soon as it exceeds one turn. It requires to keep track of the dialog state, including the history of past turns, in order to select the next appropriate step.

An SDS processes user inputs with a cascade of algorithms to extract DAs, i.e. single units of dialog moves, from speech segments [47, 120, 122]. Platforms combine agents implementing these functionalities [137, 138, 212, 213, 214].

Commercial interfaces hide the scheme of their system to protect them from being copied. The research community, through some of its members, witnessed the emergence of open-source SDS frameworks.

Among those, JUPITER [223] was one of the first SDS released to the public. The phone-based weather information conversational interface has received, between 1997 and 1999, over 30 000 calls.

Earlier, researchers from Philips [6] implemented an automatic train timetable information desk for Germany. Phone callers were initially the system's developers, then lab members. The target user group had been enlarged steadily until the service was made available to any German speaker.

More recently, Carnegie Mellon University (CMU) provided Olympus [17], "a freely available framework for research in conversational interfaces", which has been used since to build systems like RoomLine, MeetingLine, TeamTalk [74], ConQuest [16], Let's Go! Bus Information System [57, 157, 158], etc. Olympus defines a communication protocol to connect components through a single central hub which passes messages. Let's Go! Bus Information System has been deployed on a Voice over IP (VoIP) server and is answering calls since 2003. The system is able to inform users about the bus timetables and routes in the Pittsburgh area. Figure 1.3 shows the architecture of the system.

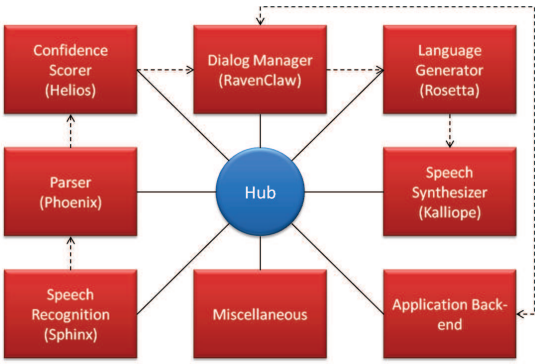


Figure 1.3: Architecture of the Let's Go! Bus Information System

In order to experiment the interaction with SDSs, a flexible modular open-

source platform has been created. The architecture is shown in Figure 1.4. It is made of a set of connected specialized services which processes the inputs of users. It has been designed to be efficient to use and at the same time simple to set up.

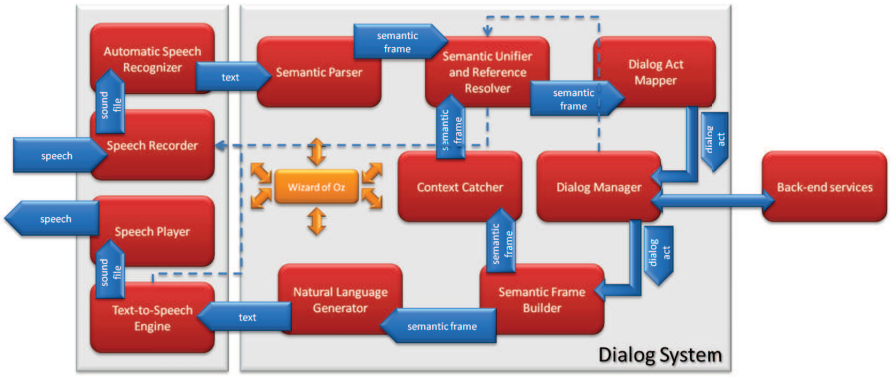


Figure 1.4: Architecture of the platform

1.2 A Step Towards Continuous Listening for Spoken Interaction

The way an SDS listens to the user has several effects on how the system handles the dialog. Current deployed commercial systems chose to give the initiative to the user to segment the signal. Most of them base that process on a binary state whose value (either listening or paused) is set by restrictive methods such as gesture recognition, keyword detection, button presses, etc. In practice, it means that every turn starts with the user signaling his/her intent to address the system before actually uttering his/her request.

The previous way of pacing the interaction is inspired by the sequential processing of dialog turns and thus it doesn't reflect the manner with which humans share information in conversations. Despite that, the reliability of such makes it an appropriate choice for SDSs, avoiding the hurdles of continuous uncontrolled listening.

When the turn-taking control is left to the machine and the capturing device is continuously recording, we talk about continuous listening. It means that the user interacts freely with the agent and that the latter segments and filters the signal. Continuous listening introduces challenges: noisy signal, speech-free segments, start and end of utterance marking, distant speech recognition, echo removal, out-of-scope utterances, etc.

This document proposes a method for continuous listening applied to the vocal interaction with a mobile companion robot. A head-mounted microphone records permanently in the home environment and the processing, based on

sound classification, parallel processing of speech, alignment scoring and attention level computation,tries to deal with the inherent issues.

1.3 A Sub-system to Map Natural-language Utterances to Situated Parametrized Dialog Acts

NLs are any languages which arise, unpremeditated, in the brains of human beings. Typically, these are the languages humans use to communicate with each other [1]. Unlike computer languages that are deterministically parsable and that obey some structural rules, NL grammars are flexible and vary from one speaker to another.

In research, NLU groups aim at extracting the meaning out of the NL input of a system, so as to build computing machines that 'understand' human language.

The modality for SDSs is speech. Thus, an NLU component's role is to analyse the (segmented) speech signal to extract meaningful information for the dialog. A unit of dialog move is called DA. It is specific to a dialog and a system, i.e. each system defines its own set of DAs and the dynamics between interaction domains. The NLU establishes a mapping between a spoken input and a DA. Note that this matching may not exists, thus a NLU sub-system needs to implement mechanisms to reject and recover from such out-of-scope utterances.

Early NLU systems based their interpretation of spoken utterances on the detection of designer-defined keywords or patterns [48, 204]. The scalability of such an approach is very limited; and so is their applicability for different domains.

Later, Context-free Grammars (CFGs) and Probabilistic Context-free Grammars (PCFGs) were applied to the understanding process [53, 68, 132, 201, 202, 203] with the aim of building parse trees covering the sequence of recognized words. This path has been taken by many who additionally proposed methods to infer grammars from data.

To avoid the burden of extracting static structures from observed data and in anticipation of the big data era, Chronus [12, 144, 145] based its processing on HMM modeling. The mapping between words in the utterances and semantic symbols was learned from data.

One last model which should be cited here is the Hidden Vector State (HVS) model [76, 77, 78, 180] which allows the creation of a parse tree from word sequences according to an automata whose history vectors are contained in HMM states. More on that will be discussed in chapter 4.

One major difference between the text extracted from Internet documents or newspapers and natural speech is the grammatical construction, which is much more flexible in the latter case. This thesis strongly argues for the partial

[1] http://en.wikipedia.org/wiki/Natural_language

parsing for SDSs in which parts of the input are processed while irrelevant ones are ignored.

A last specificity of an SDS is that, since the dialog spans several turns, the machine and the user iteratively build a shared knowledge of the interaction context and its environment. The content of that common space is reused within the same dialog and in subsequent ones, dynamically altering the meaning of utterances. The NLU components of an SDS must grab that relevant context and insert it appropriately when processing inputs.

The NLU system currently implemented within the platform performs the task of mapping user utterances to DAs according to those considerations.

1.4 The Linked-form Filling language: A New Paradigm to Create and Update Task-based Dialog Models

The DM is the decision-making component of an SDS. Based on the dialog history, the last DA and the current dialog state, it selects the most appropriate next step.

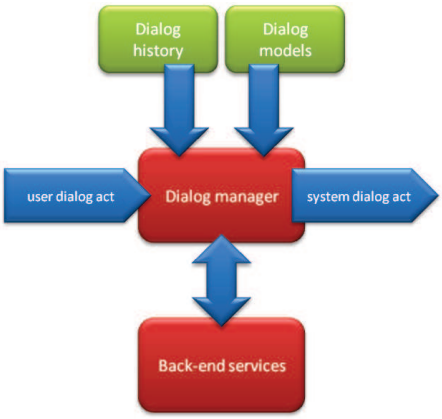


Figure 1.5: Dialog manager role

Over the years, this task has seen a variety of approaches. ELIZA [204] is considered by many as the first dialog system. Its inputs were textual messages and the adjacency-pairs analysis was the paradigm of the DM. The core of the system was built out of scripts, which associated a system's response by looking for a pattern in the input. The whole dialog was scripted that way, each script being a character for the ELIZA engine.

Larsson and Traum argued that the state of the dialog, including its history, may be represented as the sum of the information exchanged so far [101, 134]. An IS designer defines the elements of the information relevant to a dialog and

decides on their machine representation. The dialog state evolves according to a set of update rules consisting of a conditional part and an operation part. The rule applies if the condition(s), tested on the last DA and the dialog state, return(s) true. In the cases where more than one rule is available, an update strategy picks the one to be applied.

An attempt to incorporate stochastic concepts into an IS model has been proposed by Lison [115, 116, 117]. The operation part of the update rules consists of several effect, each associated with a probability value. It is computed from data and depends on the dialog state variables and the last user's input. A Bayesian network encodes the dialog state. The rules update the Bayesian network's variables that encode the dialog state according to their effects and, based on the new stochastic dialog state, decide on the most appropriate next step.

An example-based DM [91, 106, 108, 109] constructs a request to a database from the annotated input DA (primary DA, discourse DA, history vector). The database stores examples seen in interaction data. The algorithm looks for the most similar entry in the base and executes the associated system's action.

Work flows have been applied to DM. Those automata encode the system response within the states while the transitions are deterministically extracted from the user's turns. This may be the simplest way to model dialogs but it lacks flexibility and naturalness.

The increasing amount of available interaction data favores the development of stochastic systems. The MDPs are appropriate to manage dialogs [113, 114, 167]. In those, the dialog state space contains all the states the dialog may be in and the transitions depend on the user inputs. The behaviour of a DM based on MDPs is defined by a strategy which associates to each states an action to be executed. This strategy is learned from data so the designer only needs to define the domain.

POMDPs extended the MDPs hiding the states which emit observations according to a probabilistic distribution [25, 28, 79, 94, 167, 191, 207, 218, 221]. This additional layer encodes the uncertainty about both, in the case of SDSs, the ASR and the NLU. Currently, practical POMDP-based DMs are limited in the number of variables and by the intractability of the optimal strategy exact computation [24, 81, 119, 135, 150, 190, 206, 208, 209]. Methods have been proposed to reduce the search space [24, 81, 119, 135, 150, 190, 206, 208, 209, 220], others explores the use of a simulated user to efficiently collect interaction data [31, 32, 54, 55, 93, 111, 146, 166].

This thesis chose to use task hierarchies to model the dialogs [18, 20, 72, 112, 161, 162, 163, 164, 165, 182]. A task-based manager applies the following rule: a task is achieved when all its children tasks, if it has any, are achieved, by itself otherwise. In other words, a task hierarchy defines primitive tasks and intermediate nodes between them to create dialogs.

Such a paradigm allows for a direct update of the models since it only takes changing the branching or substituting primitive tasks to do so. Also, there is no need for additional data. The hierarchy enables users to utter extra information, which are looked for to be grounded in a limited part of the trees.

However, building a task model and updating and maintaining it is increasingly difficult proportionally to its size and complexity of the desired behavior.

This thesis proposes a new paradigm to shift the modeling task to an application specification one. A designer willing to write a dialog model for his/her application/service describes the specifications in the LFF language. This code is transformed automatically to a task-based dialog model, compliant to the ANSI/CEA-2018 standard [41]. Thus the complexity of the low-level hierarchy is hidden and the design is facilitated.

The following document is divided into five chapters. At first, the open-source platform is introduced, its architecture and its processing flow. According to the order in which an input is processed, the second chapter deals with the listening control, tightly correlated with the third one whose topic is the personalization of SDSs. The NLU sub-system is described next and the final part of the thesis is about the dialog modeling using the LFF language.

Chapter 2

A Modular Open-source Platform for Spoken Dialog Systems

2.1 Introduction

The central theme of this document is a modular open-source platform for SDSs. This chapter introduces it to the reader.

2.1.1 SDS Definition

An SDS is defined as a system providing an interface to a service, may it be virtual or concrete, via a human-like dialog [35]. An interaction qualifies as a dialog as soon as it exceeds one turn. It requires to keep track of the dialog state, including the history of past turns, in order to select the next appropriate step (cf. 5.2).

The SDS worflow is usually split into distinctive features which recognize the speech, extract the information relevant to the dialog, act on back-end services, decide on the best next step, generate NL from DAs and synthesize speech.

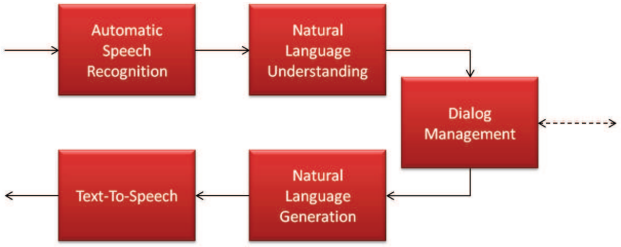


Figure 2.1: State-of-the-art SDS's chained processes

In this chapter, the modular open-source platform for SDSs is introduced. An overview of it is provided along with an example of interaction to detail the functionalities of each component. The components which define the outgoing interface are described while the others are explored in other chapters. Experiments with the platform are reported here as well.

2.1.2 Deployed Research Systems

One can distinguish two main directions in the current status of SDS development. On the one hand, global technology companies such as Apple, Google, Samsung, etc, recently released their virtual speaking assistants for hand-held devices. The competition between those entities is tough. Thus, the inner methods are kept hidden and the systems are based on protected remote servers. Research open systems generally suffer from the lack of in-domain data and potential end users.

In this section, some research systems are presented followed with insights on the current gap between commercial SDS and them.

Olympus (Let's Go)

The Let's Go! Bus Information System [57, 157, 158] is the result of a collaboration between CMU and the Pittsburgh's Port Authority. The system has been deployed on a VoIP server and is answering calls since 2003. It is able to inform users about the bus timetables and routes in the Pittsburgh area. Thus it interacts with a broad range of callers with varying knowledge about he system's abilities, its vocabulary set, etc.

The training and configuration data was provided by the Port Authority of Allegheny County. It consisted of the bus schedule and recorded dialogs between customers and human operators.

Architecture The Let's Go! Bus Information System is based on the Olympus architecture [17], "a freely available framework for research in conversational interfaces", which has been used to build systems like RoomLine, MeetingLine, TeamTalk [74], ConQuest [16], etc. Olympus defines a communication protocol to connect components through a single central hub which passes messages. Figure 2.2 shows the architecture of the system.

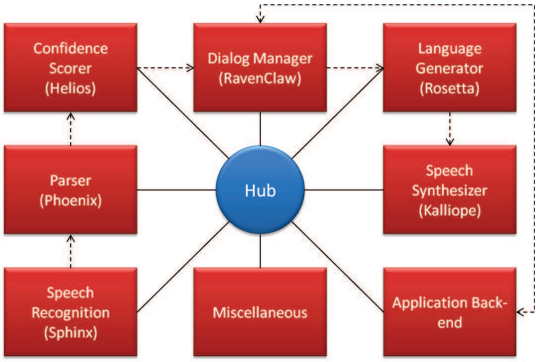


Figure 2.2: Architecture of the Let's Go! Bus Information System

Speech recognition The speech recognition is achieved with Sphinx [198]. The system runs two engines set up with different gender-dependent Acoustic Models (AMs) and a shared LM.

Natural language understanding Phoenix [19, 202], which parses the user's utterances is based on a CFG. In addition, a component, Helios [16, 17, 142, 156], annotates the inputs with a score computed by a logistic regression model-based algorithm.

Dialog management RavenClaw [18] is the task-based DM of the bus information system. The engine is a task-independent inference engine that is configured with task hierarchies.

Natural language generation and speech synthesis Rosetta and the Kalliope speech synthesizer are combined to generate the natural-language spoken outputs of the system.

The deployment of the Let's Go system to the general population and the release of the collected data make it a great tool for evaluating, testing and training SDSs.

JUPITER

Jupiter [223] is a phone-connected SDS to get information about the weather forecast in selected cities in the United States [223]. It is based on the GALAXY-II architecture [179].

Architecture The JUPITER's architecture is shown in Figure 2.3.

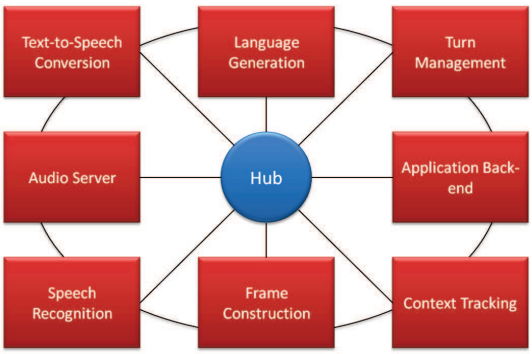


Figure 2.3: Jupiter's architecture (source: [223])

Speech recognition The SUMMIT ASR engine is set to recognize 1957 words including 650 cities and 166 countries using landmark-based diphone acoustic modeling, a lexicon and bigram or trigram LMs.

Natural language understanding The NLU in the system is used in two ways. TINA:

- parses the user's queries to extract the intent of the caller.
- analyses the retrieved weather forecast reports to convert them to frames that the NLG module is able to process.

Dialog management Called turn manager in the architecture, the DM defines a set of operations that are executed whenever their pre-conditions are fulfilled.

Natural language generation Text messages intended to be synthesized to the user are generated by the GENESIS engine based on rewrite rules, a lexicon and templates.

The Philips automatic train timetable information system

The Philips automatic train timetable information system went public in 1995 [6]. It allows users to make enquiries to the national German train timetable in a naturally speaking manner.

Architecture The Philips automatic train timetable information system's architecture is shown in Figure 2.4.

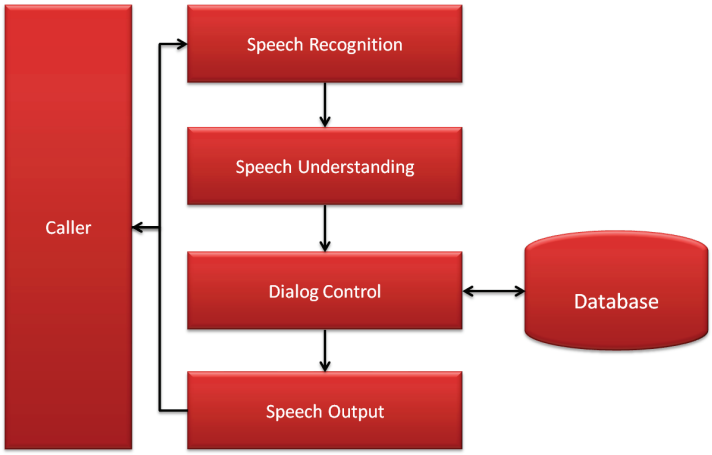


Figure 2.4: Philips automatic train timetable information system's architecture (source: [6])

Speech recognition The ASR task is carried out by the PHICOS system, a research product of the same company. The acoustic models are 6-states left-to-right HMMs of 40 phonemes. The total number of words the module can recognize is 1850. 1200 of those lexicon entries are train station names.

Natural language understanding The SDS applies PCFG rules to parse the user utterances. Special tokens such as filler words and silences have been integrated in the grammar. The parse trees are then extended to associate a database's search slot with semantic concepts.

Dialog management The DM handles the database's requests to add/relax constraints when needed. It produces the text segments to interpret the results of the queries or asks the user for more information.

RailTel, Arise, Ritel

In 1997, the RailTel project [99] studied the technical adequacy of vocal telephone services to get rail travel information. The RailTel system, whose architecture is shown in Figure 2.5, has been the baseline for the Arise [100] SDS (Figure 2.6).

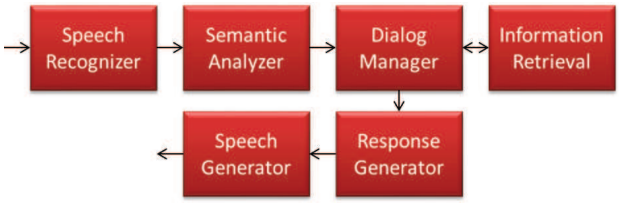


Figure 2.5: RailTel system's architecture (source: [99])

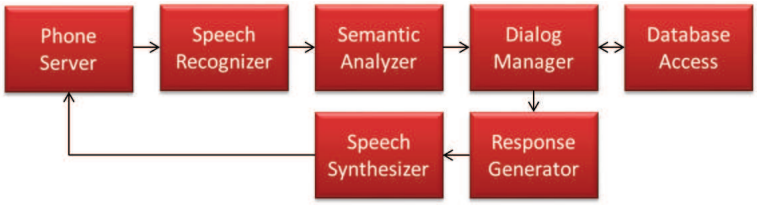


Figure 2.6: Arise system's architecture (source: [100])

The Ritel system [61] proposes to merge a question-answering system and an SDS. The goal is for the system to be able to answer general questions with an intelligent refinement stage when necessary. The system is phone-based as shown in Figure 2.7.

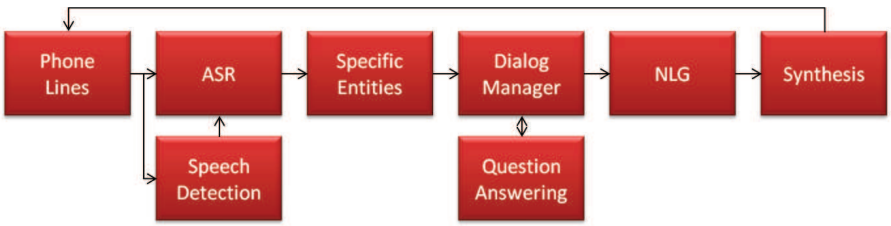


Figure 2.7: Ritel system's architecture (source: [61])

WikiTalk

Another open-domain SDS worth mentioning is WikiTalk [4, 44, 89, 205]. This program claims to be an open-domain knowledge access system. Indeed, it uses Wikipedia as a very large base of topics to talk about. However, unlike many similar systems, WikiTalk uses a finite-state machine to keep track of the current topic and manage the shift from one to another.

The advantages of such an approach is in the huge quantity of Wikipedia entries available and the human-annotated hyperlinks that are used to implement smooth topic shifts. Moreover, the vocabulary set that may be uttered by the user – and thus the one that needs to be recognized – is dynamically

adapted with the latent user topic shifts, called NewInfos, and a limited set of commands such as "start", "stop", "continue", etc. This improves the accuracy of the speech recognizer before applying keywords spotting methods to the transcribed user utterances.

The finite-state machine to manage topics in WikiTalk is shown in figure 2.8

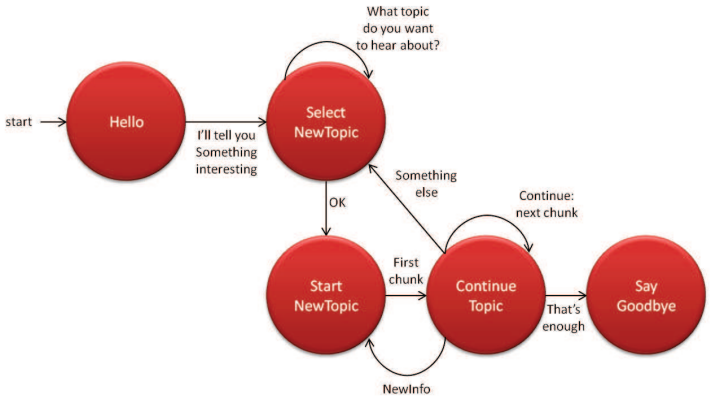


Figure 2.8: WikiTalk's finite-state machine for topic tracking (source: [205])

2.1.3 Commercial and Research Perspectives

Lately speech and other types of NL are experiencing an increased acceptance when being used for interacting with "intelligent" computing systems. Companies increasingly provide us with potentially new application scenarios where the modality is seen as the best way of operation. This trend is particularly reflected by recent technology releases such as Apple's Siri, Google's Google Now, Microsoft's Cortana, Samsung's S Voice and LG's Voice Mate.

While these products clearly demonstrate the industry's vision of how we should be interacting with our current and future devices, they also highlight some of the great challenges that still remain [140]. Indeed, criticisms have been received questioning the reliability, the usefulness, the data protection, and the proprietary aspects of the technologies, etc.

NL interaction is, despite those recent advances, still not reliable enough to be used by the majority of users and hence hardly accepted as an efficient way to communicate with a machine. We face a socio-technological problem where the use of those error-prone technologies may easily lead to unsatisfying user experiences. While for some talking to a computer may simply convey a great user experience, for others, it can offer a significant alleviation when interacting with a piece of technology. However, the leap forward taken by ASR has demonstrated how a technology entering the virtuous circle of machine learning Figure 2.9) may significantly improve its performance.

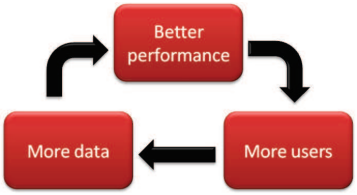


Figure 2.9: Virtuous circle of machine learning

SDSs depend on the technological advancements of those components which essentially define them. They aim at mimicking the behaviour of a human supporting the spoken interaction. The cognitive abilities that we, humans, can demonstrate are complex in many ways. Thus current systems, even the mosts advanced ones struggle to act like true autonomous-thinking entities, and consequently appear to the user as unnatural. Some challenges to be tackled are the turn taking, the DA extraction from speech, context inclusion, decision making, adaptive learning, or personalization.

Commercial interfaces hide the scheme of their system to protect them from being copied. In the research community, however, we have seen the emergence of open-source frameworks.

This chapter serves as the primary documentation for an SDS platform that integrates and extends some of these existing open-source components. The modular architecture of the SDS is explained with a running interaction to describe the functionalities of every component. The listening control, the ASR, the NLU and the DM are the topics of dedicated chapters and thus will not be explored here. However, the speech synthesis and language generation components are detailed along with a technical point of view on the speech recorder and player. The experiments conducted with the platform conclude the chapter. Their goal was both to collect training data and to evaluate the overall system with real users.

2.2 A New SDS Platform

Here, an overview of the platform is presented.

2.2.1 Desired characteristics

To start with, a list of the characteristics that a new SDS platform should have has been established. This was defined from both the research projects requirements and the study of the state of the art in the domain.

According to that list, proposing a new SDS platform is relevant if it brings the features described next.

- Server based: a server-based system is able to manage many users concurrently and remotely.

- Extendable: the platform architecture is flexible and may evolve. It does not prevent the integration of additional components, resources or services.
- Modular: any component making up the platform may be isolated, replaced or, if not essential, removed.
- Multi-lingual: the platform may be set for many languages. It is not constrained to the original one(s).
- Multi-modal: even though the primary objective is to build a spoken human-machine interface, the platform is sufficiently generic to allow for the plug of some other modality sensors/processors.
- Open-source: the platform sources are available for use and modification.
- Facilitated setup: the platform main objective is to facilitate the access to full dialog systems to specialists of narrower domains such as the ASR, the NLU, the NLG, the DM, etc. These non-experts developers needs facilitated methods to set up the platform to their convenience.
- Minimum data requirement: the advent of the machine learning era leads to believe that a computing machine may do almost anything given that it has access to enough data to train with. The platform allows for skipping most of the data collection beforehand. One may set up a first working system which then can be used to collect training data (see figure 2.9).

Table 2.1 shows the characteristic mapping of several platform and systems.

	Server-based	Modular	Extendable	Open-source	Minimum data requirement	Multi-lingual	Multi-modal	Facilitated setup
CLSU Toolkit [189]	×	✓	✓	✓	✓	✓	×	✓
GALAXY-II [179]	✓	×	×	✓	✓	✓	×	×
Voice XML	✓	×	×	✓	✓	✓	×	✓
Olympus [17]	✓	✓	✓	✓	✓	✓	×	×
OpenDial[1] [115]	✓	✓	✓	✓	✓	✓	×	✓
Siri-like systems	✓	-	×	×	-	✓	×	-

Table 2.1: Characteristics mapping for SDS platforms
✓: has the characteristic
×: does not have the characteristic
-: the information is not available

2.2.2 Architecture

The usual chained workflow of an SDS is represented in Figure 2.10.

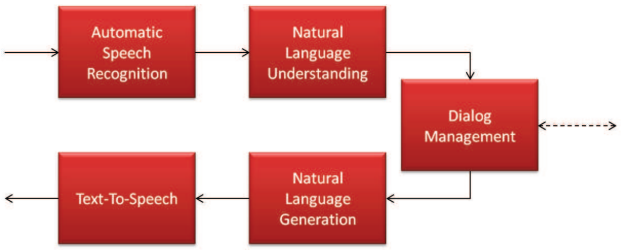


Figure 2.10: State-of-the-art SDS's chained processes

It is a uni-directional sequence of processes whose turning point is the DM. ASR and NLU interpret the user's spoken inputs and generate natural-language system's utterances from formal DM data structures (or DAs).

The SDS built in this work extends this design. Components are split into modified sub-modules and new processes are integrated to the state-of-the-art workflow chain. Figure 2.11 shows the current status of the SDS's implementation. It consists of a set of connected specialized services which process the inputs of users. It has been designed to be efficient to use and at the same time simple to set up.

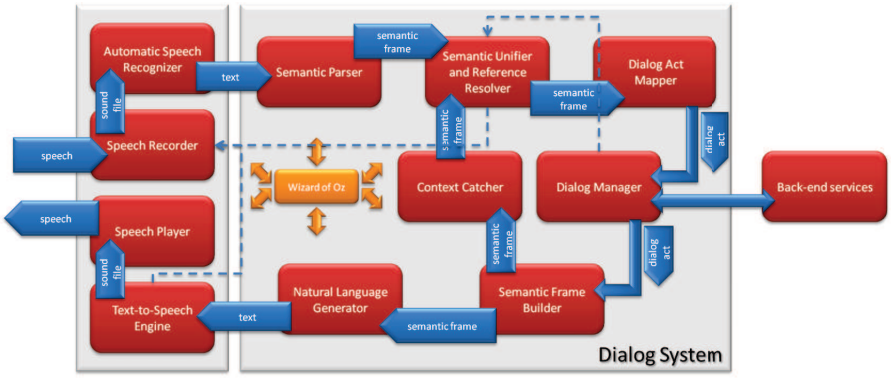


Figure 2.11: Architecture of the platform

2.2.3 Communication

All the platform's components communicate with ActiveMQ[2], a message queuing protocol available for Java, C and other programming languages. Ac-

[1] As of version 0.95, released on April 4, 2014
[2] http://activemq.apache.org

tiveMQ's architecture is built around a broker whose task is to connect the registered clients, collect messages and distribute them to receivers of topics and queues.

This protocol is standard, light and flexible. It allows for the substitution of components and their implementation as independent services.

2.2.4 Grounding

Since an SDS is an interface between users and applications, it needs to provide methods to gain access to various background services. Simply sustaining a dialog is the characteristic of a conversational agent. An SDS, on the other hand, defines a goal, which should be reached in the most efficient way.

Grounding is the process of communicating between the SDS and the back-end services, i.e. linking the information representation within the DM and the API of the services [38, 39].

The DM detailed here currently implements anytime-calls to back-end software. The grounding process is done via ECMAScript pieces of code calling external APIs.

The grounding is usually most effective at the end of a dialog, i.e. once enough necessary pieces of information have been collected. However, in order to synchronize the SDS with a graphical application, some more message exchanges have been integrated, so as to update the remote client in real time with the spoken data collected (to be displayed).

2.3 Interaction Example

In this section, the functionalities of the platform's components are described. It is based on an interaction example whose in-depth system's processing is observed step by step. The reader is invited to follow the workflow in the architecture schematic above Figure 2.11).

2.3.1 Simulated Service Description

The service the SDS mediates is the access to a weather forecast information system (similar to JUPITER). The necessary pieces of information to start a search request are the location, e.g. the name of a city, and the day one wants to get the forecast for. We assume here, for the sake of this example, that the system can understand any location in the world and that the forecast is known in advance for an (unrealistic) infinite period of time. The user-defined detail level is binary. When high, the system returns the wind speed, the temperature, the air humidity, the likelihood of rain and the UV index while a low level causes the system to summarize the forecast to a shallow overview.

2.3.2 One Interaction Turn

1. The system is idle. The user has connected to the SDS via a voice client but has not spoken yet. We assume here that the first signal segment perceived by the system contains only speech[3].

2. The user addresses the system: "What is the weather forecast for today?". He/she did not have to signal his/her intent of speaking before talking. The system handles the listening automatically, as the sensor records the signal continuously.

3. The continuous signal entering the segmenter is split according to a silence delay. The segments start when a signal frame's amplitude exceeds the silence threshold. The component ends a segment when it detects that the signal's amplitude has been below the same threshold for a certain duration of time. Both the threshold and the delay are dynamic parameters that can be set off-line as well as in real time.

4. A segment is rejected if its duration is less than the minimum length parameter. Otherwise, it is sent, through ActiveMQ, to the ASR input queue associated with a user Identifier (ID) and a language code.

5. The ASR service provides up to seven ranked hypotheses about the content of the signal segment. The confidence score of the best hypothesis is returned as well. Non-speech segments do not allow for this score to be high enough to be considered as relevant. Thus, the set of hypotheses one obtains is empty. The ASR component provides, in this case, a default utterance "unknown". All hypotheses are passed on to the next module, whether the set consists of seven, or less, or a single "unknown" hypothesis.

6. The Semantic Parser (SP) aims at labeling the transcriptions with semantic concepts. Such concepts represent the meaning of the combined words. The SP in the platform tries a sequence of transformation rules. Whenever the applicability conditions are true, the transformation is applied. The algorithm bases the extraction of the concepts on the transcriptions only, without any knowledge of the current dialog state or the application environment. The data structures produced are Semantic Frames (SFs) (one for each hypothesis) made of a goal and zero or more slots.

7. The Semantic Unifier and Reference Resolver (SURR) information base consists of trees of semantic concepts. The algorithm searches for a path in the graph to reach root nodes from SFs' slots and goals. The component has several usages:

   (a) It is used to resolve relative references. In that example, "today" is a dynamic concept, i.e. it changes every day, that does not link to an absolute date. The SURR has external calling nodes whose inner

[3]See chapter 3 for non-speech signal processing

value(s) is(are) set by accessing some host functionalities or APIs. One of them gives an absolute value to the "today" concept. The component requests the current date of the host system, formats it and substitutes the values. The current time and location can be obtained the same way.

(b) It is used to solve dialog contextual references. The Context Catcher (CC) maintains the SURR up to date with the current status of the dialog, modifying the nodes and the branching of the latter's inner forest. That way, the dialog context is injected into the SFs at the SURR stage.

(c) It is used to adjust the semantic level of the analysis. The SP tries to label the utterances with semantic concepts which may not be included in the semantic concept set of the Dialog Act Mapper (DAM), whose semantic space is less large. The SURR bridges this gap merging/converting/splitting the SF slots to a level characterized as root, using the same tree representation described before.

(d) It manages the hypotheses set. For a single interaction turn, the SURR gets all the SF hypotheses derived from the seven (or less) ASR transcriptions. Only the best one is passed on to the DAM after being processed by the SURR. The others are sent after the DAM, when unable to map the SF to a DA, requests the next one in the ranked list. If the bottom of the set is reached the best hypothesis is sent again with a "final" flag.

Thus, out of this component, one gets an absolute contextualized situated SF for the DAM to process.

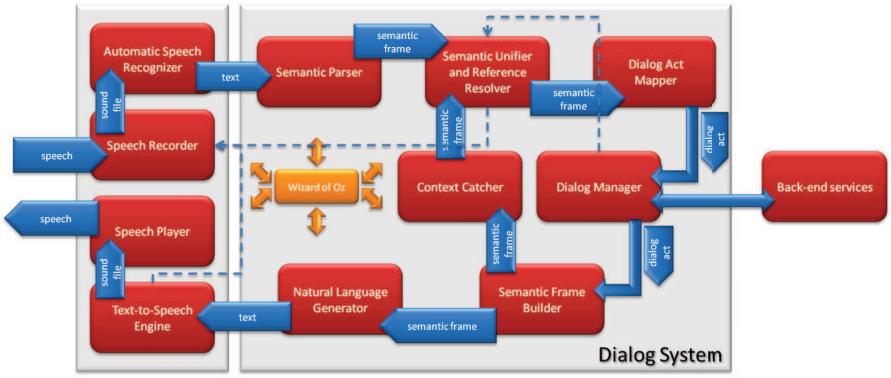


Figure 2.12: Architecture of the platform

8. The ultimate NLU component retrieves the set of DAs available in the current dialog state, then it tries to map the received SF to one of them. The mapping is actually a conversion from goal to DA intent and from

slots to parameters. The difficult task, however, is to get the current set of DAs. The DAM has to explore the stack of tasks, the partially elaborated plans and anticipate the likely next turns looking for a variable already mentioned in the user speech. If the mapping can not be found, the module requests the next hypothesis from the SURR unless the final flag is true.

9. The DM consults the dialog history and task models, and associates the last DA to update the dialog state, possibly connecting to back-end services and selecting the most appropriate next step to apply. The models define the dynamic sequence of those to be executed in order to access the service in the most efficient way. The DM works with the units of dialog called DAs, it gets some as inputs and sends some out.

10. The DAs out of the DM are projected back into the semantic space, the system's intent becomes the goal of the SFs and the parameters are converted to slots.

11. The NLG is set with templates to express the parametric meaning represented as an SF. It matches the input with the templates in an SF-constrained search . If more than one template has the same characteristics, a random selection picks the one to be instantiated. More on this process will be discussed in section 2.5

12. The speech synthesizer generates speech from text. It is based on a remote server computation. The text is embedded in an Hypertext Transfer Protocol (HTTP) request and the signal stream is captured back and stored in a file. Parameters are the text, the language and the voice one wants to synthesize.

13. The generated speech is played back to the user: "Which city do you want the forecast for?"

2.3.3 Interaction Turns

Here is the whole interaction with the inter-component messages.

From the user to the Speech Recorder "What is the weather forecast for today"

From the Speech Recorder to the ASR en /tmp/6002-2014.1001.151252.flac

From the ASR to the SP en 6002-2014.1001.151252;what is the weather forecast for today;who what is the weather forecast for today;what is the weather forecast today;who what is the weather forecast today;

From the SP to the SURR en 6002-2014.1001.151252;goal=input; freetext=what is the weather forecast for today;goal=request_forecast;date=today; goal=request_forecast;date=today;goal=request_forecast;date=today; goal=request_forecast;date=today;

From the SURR to the DM en 6002-request_forecast:date=March 11, 2015;reco:what is the weather forecast for today

From the DM to the NLG en 6002-Ask.What:goal=get_location;slot=location;

From the NLG to the TTS en 6002-which city do you want the forecast of?

From the TTs to the user "Which city do you want the forecast of?"

From the user to the Speech Recorder "For Paris"

From the Speech Recorder to the ASR en /tmp/6002-2014.1001.151301.flac

From the ASR to the SP en 6002-2014.1001.151301;for paris;

From the SP to the SURR en 6002-2014.1001.151301;goal=input;freetext=for paris;goal=input;city=paris;

From the SURR to the DM en 6002-input:location=paris;reco:for paris

From the DM to the NLG en 6002-Ask.What:goal=get_detail_level;slot=detail_level;

From the NLG to the TTS en 6002-which level of detail do you want?

From the TTs to the user "Which level of detail do you want?"

From the user to the Speech Recorder "The lowest one"

From the Speech Recorder to the ASR en /tmp/6002-2014.1001.151307.flac

From the ASR to the SP en 6002-2014.1001.151307;the lowest one;

From the SP to the SURR en 6002-2014.1001.151307;goal=input;freetext=the lowest one;goal=input;level=low;

From the SURR to the DM en 6002-input:detail_level=low;reco:the lowest one

From the DM to the NLG en 6002-Feedback:goal=report;content=it is sunny in paris, the temperature is 25;

From the NLG to the TTS en 6002-it is sunny in paris, the temperature is 25

From the TTs to the user "It is sunny in paris, the temperature is 25"

2.4 Speech synthesis

MaryTTS [141, 177] is an open-source speech synthesis framework maintained by the Cluster of Excellence MMCI and the DFKI. It offers pre-built voice models for different languages as well as tools to create and manipulate them.

The MaryTTS module is a client to a generating server, which may be hosted on the same machine. A request containing the text to be synthesized with additional prosodic information is sent to the central server which returns the speech stream. The TTS module of the present platform is a basic client program embedded into an ActiveMQ wrapper.

2.5 Natural Language Generation

The NLG component is based on sentence templates to be instantiated. The inputs are SFs from the DM and the outputs are system's text utterances.

NLG is an active research area in the Human-Computer Interaction (HCI) domain. In our case, a simple but effective solution to produce NL utterances conveying the DM's messages was targeted. Received messages are SFs built into the interface converting the DM's outputs. The engine is fed with a set of templates made of a title (identical to an SF's goal) associated with an utterance whose parts may be replaced by slot names or slot name-value pairs. Table 2.2 presents the 3 patterns one can find in the configuration file of the NLG and their respective instance example.

[...]
greetings: good morning. How are you?
"Good morning. How are you?"
personalGreetings: good morning /firstname/.
"Good morning John"
situatedGreetings: good evening [/time=evening/].
"Good evening"
[...]

Table 2.2: NLG template file sample (and instances examples)

The selection algorithm first compares the goal of the input SF with the titles and marks the matching templates. Then a second pass removes the ones requiring slot names or slot pairs that are not contained in the input SF. At this stage, if there are still more than one template available, the algorithm randomly picks one of them, substitutes the slot's markers with their values defined in the SF, and discards the slots in squared brackets. The result is a NL utterance to be synthesized or displayed on a screen.

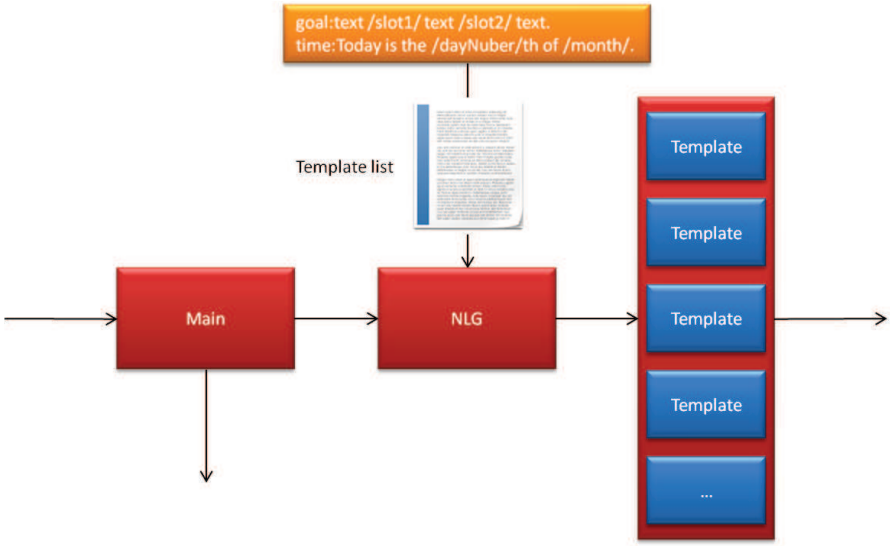


Figure 2.13: Schematic of the NLG process

2.6 Speech Recognition

The system has been built with two different ASR engines. The first integrates the Julius [104, 105] recognition engine, the second uses the Google Speech API.

2.6.1 Local Implementation

Julius, developed at the Kawahara Lab at Kyoto University, was our first choice as a Speech recognition engine. It is able to process large-vocabulary search in real time, running a 2-pass algorithm. The configuration of the engine consists of 3 Knowledge Sources (KSs): an n-gram LM, a set of acoustic HMMs and a lexicon. In addition, Julius parameterizes digital speech segments on the fly according to an HTK-compliant parameter set. The engine can concurrently process identical audio segments with several instances. All those features made it the appropriate automatic speech recognizer for our platform.

The phonemic HMMs were trained on the ESTER and ETAPE [62] corpora

which are based on transcribed French radio and TV broadcast. The LMs are learned from the text resource from "Le Monde".

2.6.2 Web Service

For the web-based implementation of the Google Speech API, the service specification are the following. An HTTP POST request transmits the signal segment to be recognized. The API returns the n-best hypotheses, where n is a parameter of the request, and the confidence score of the best one. An empty result is returned when the speech segment cannot be recognized with enough confidence, i.e. it does not contain speech.

2.6.3 Speech Recognizers Benchmarking

The history of the lab within which this work has been achieved is mainly about signal processing, especially speech recognition and biosignal processing. Over the last decade, the most relevant pieces of software that were both set up and with which people had experience with were the HTK and Julius. AMs and LMs have been built for many languages such as Spanish, German, Dutch, French, English, Italian, etc.

Earlier, the trade-off in the speech recognition task was mentioned. The larger the domain, in terms of vocabulary size and speaker variability, the lower the performances of a speech recognition system. Consequently, the smaller it is, the more reliable one can expect the produced hypotheses to be.

Also, ASR engines, when applying machine learning methods such as the HMMs or the neural networks, require as much data as it is possible to gather and process. In regards to that, the Google global company possesses, obviously, much more data than is available in the research lab. They have the tools, the man power and the computing ability to, as this report is being written, collect utterances from speakers all over the world.

The benchmarking in this section offers subjective measurements to answer the question: "What ASR should I use for my SDS?". This topic has been investigated in [133]. Here is a comparative evaluation of a Julius-based speech recognizer and the Google Speech API using the data collected with the SDS platform in the vAssist project.

Corpus The speech data, collected in the real-user experiments (cf. Section 2.7), has been manually processed offline. Since the collection phases used the Google Speech API to transcribe speech, the transcription of every entry in the database (see Table 2.5) has been checked and/or corrected by a human annotator.

Pairs *(speech file, transcription)* have been extracted to build a reference corpus. It contains 1037 speech segments, whose length varies between 1 and 24 words. Overall, there are 8618 words in the corpus. Speech files are encoded on 16 bits at 16 kHz as Signed Integer PCM. Although the recording environment was free of ambient noise, the automatic segmentation as well as the unintended

utterances, i.e. those that are not addressing the system, were kept so that the actual SDS usage conditions are preserved for the experiment.

Method Figure 2.14 shows the method of evaluation.

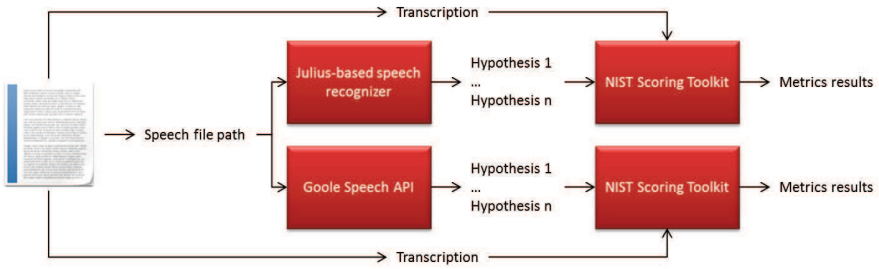


Figure 2.14: Benchmarking setup

Each corpus' speech segment was submitted to both modules, which attempted to produce up to five hypotheses on the transcription. These are compared to the reference transcriptions using the sclite tool from the NIST Scoring Toolkit (SCTK).

The comparison is performed in three ways.

At first, all hypotheses from a single speech segment are aligned with the reference transcription. The metrics are averaged over all the hypotheses. This method is used to give a primary overall performance measure of an ASR system.

The first hypothesis is the top ranked one out of a speech recognizer, i.e. it is the one which the module gives the best confidence score for. In an SDS, this first hypothesis is the one that is the most likely to enter the dialog flow. Indeed, some systems are only processing a single top-ranked word hypothesis resulting from the speech recognizer.

The here presented platform has the ability to handle multiple hypotheses. Then, the most relevant measure, in this framework, may be the best hypothesis comparison. The best hypothesis is the one that gives the best match compared to the reference sentence. Again, all hypotheses from a single speech segment, are aligned with the reference. Only the best score is added to the computation of the averaged measures.

Metrics Two metrics are computed. The sentence error rate is measured, as well as the word error rate.

$$sentence\ error\ rate = 100 \times \frac{count\ of\ incorrect\ hypothesis\ sentences}{count\ of\ reference\ sentences}$$

$$word\ error\ rate = 100 \times \frac{count\ of\ incorrect\ hypothesis\ words}{count\ of\ reference\ words}$$

Results The following table shows the averaged metrics' values for the three different analyses.

		Julius-based speech recognizer	Google Speech API
All hypotheses	Sentence error rate	97.9%	88.9%
	Word error rate	58.1%	28.1%
First hypothesis	Sentence error rate	94.9%	95%
	Word error rate	56.9%	28.8%
Best hypothesis	Sentence error rate	94.2%	56%
	Word error rate	55.7%	21.5%

Table 2.3: Comparative evaluation of the speech recognizers

Conclusion This benchmarking demonstrates that the Google Speech API is more reliable than the open-source version available in the lab. The web service covers the vAssist domain. The continued focus on that technology from the global company lead one to expect the performance to improve with time.

However, the Google Speech API does not allow for any modification of the recognizer KSs. In the next chapter (Chapter 3), two listening methods are presented. One is based on the adaptation of the KSs to the user, the environment and the dialog context while the other uses the Google Speech API and thus can not be dynamically updated.

2.7 Real-user Data Collection

Real users have been asked to interact with the SDS. There are three motivations for that:

- Real data related to the actual domain of the application have to be collected and annotated in order to train some modules.
- A user-interaction corpus is the basis to evaluate individual components as well as to provide a benchmark of comparable performance measures to monitor the system iteratively.
- The development of the platform is based on a user-centered design method.

Experiments were conducted in Vienna and Paris to collect data, evaluate the usability of the system and elicit recommendations for later development.

2.7.1 Wizard of Oz

A Wizard of Oz (WoZ) tool is a quick prototyping/evaluation method. It consists of a human operator replacing part(s) or the whole of an automatic system to collect data, get user feedback and evaluate individual components and usability at an early development stage.

2.7.2 WoZ-based Lab trials

A first data collection session took place in Fall 2013 in Vienna and in Paris with elderly people, which are the target group members. Within the first lab evaluation a mix of different qualitative and quantitative measurements and the WoZ method were applied to gather first insights about usability and interaction aspects and to collect a first set of natural speech data that will feed the development process of the speech-based interaction concept of vAssist. This section provides a detailed overview of the Austrian and French lab trial results that constitute the basis for recommendations and system interaction improvements.

Figure 2.15 shows example screenshots for the two main GUIs, which are the PillBox and the DailyCare applications.

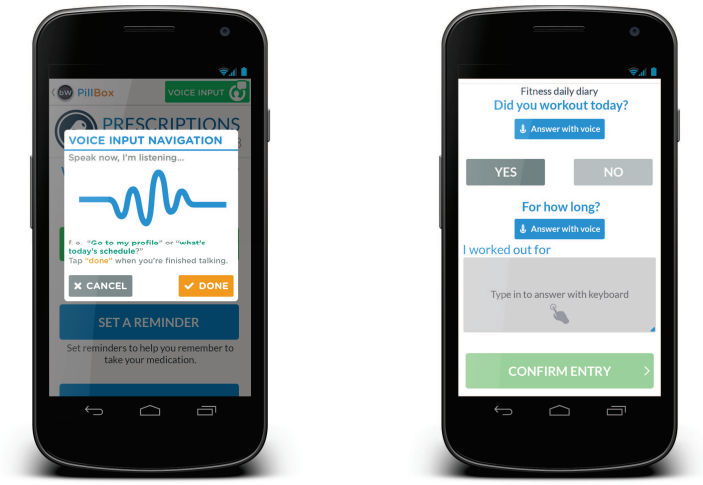


Figure 2.15: Screenshots from the PillBox (left) and the DailyCare (right) applications

Setup

The trialed system is different in the first experimentation phase and in the second one. Indeed, the data collected are different for each one of them.

Figure 2.16 shows the first SDS data collection setup.

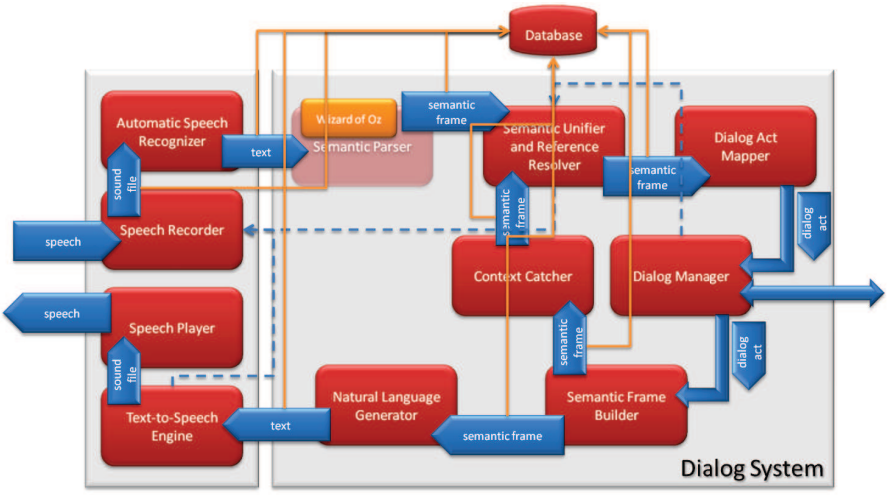


Figure 2.16: First setup

The location was the experience room of the lab. The noise was limited to the one produced by people inside the room. The user was facing the laptop displaying the press-to-speak button and integrating the microphone and two speakers. The facilitator was seated next to the experimenter. The WoZ operator, who was outside the room received the user's utterances as transcribed by the ASR service and created on the fly the SF expressing the same intent. This data structure was sent back to the automatic system, which contextualized SFs, managed the dialog, and generated and synthesized the responses.

Users were introduced to the system, mentioning that it was fully automatic, and then asked to perform a number of interactions with it. The inter-component data was recorded as shown in Figure 2.16. The goal of this phase was mostly to collect data to train the NLU sub-system.

Users

In total 15 persons participated in the first lab trial; 7 (one drop out) in Austria and 8 in France.

Table 2.4 below provides an overview on the sample, demographic data and pre-interview results referring to experience with technology; physical constrains that might affect technology interaction; applied coping strategies; doctor visits and the perceived stress level of doctor visits.

Data	Austria	France
Participants	7	8
Age (range)	66.6 (60-72)	77.0 (65-86)
Gender	4 ♀, 3 ♂	7 ♀, 1 ♂
Retiree	6/7	8/8
Experience with touchscreen devices	3/7 have experience with touch-screen devices	5/8 have experience with touch-screen devices
Physical constraints	3/7 fine motor restrictions	4/8 chronic disease
Coping strategies	Pencils, integrated extendable keyboard	Big size screen
Frequency doctor visits	3/7 less than once a month, 4/7 more frequent	8/8 less than once a month
Perceived stress level	On average not very stressful	7/8 not stressful at all
Reduction of doctor visits requested	5/7 yes, 1/7 ok, 1/7 no	3/8 ok, 5/8 no

Table 2.4: Second lab trials user sample description and pre-interview results

Collected data and results

The collected data has been aligned and combined so that each entry in the database consists of the sound file's absolute path, the best ASR hypothesis, the WoZ-generated semantic parse, the semantic frame out of the NLU subsystem, the system's DA and the CC messages (if any). The ASR hypotheses have been manually checked offline.

Speech file's absolute path	*/home/xxx/recordings/ 2013.0910.153845.wav*
Best ASR hypothesis	*La première prise est à 9h30*
Semantic parse	*input:first_intake=9h30*
Semantic frame	*input:first_intake=9h30*
System's DA	*Ask.What:slot=confirmation*
CC messages out	*retract:rewrite(frame(G, S), frame(G, [slot(affirmative, V)])) asserta:rewrite(frame(G, [slot(confirmation, V)]), frame(G, [slot(affirmative, V)]))*

Table 2.5: An entry of the database obtained after the first data collection session

The database contains 75 dialogs consisting of 336 turns. That is less than 12 minutes of cumulated speech data.

The data collected allows for training the SP. The corpus contains the text utterance of the users and their WoZ-generated SF.

Moreover, the user feedbacks (cf. Section 3.9.4) highlighted some expectations and consequently guided the next iterations of development.

2.7.3 System Trials

The second user experiment session was achieved using a fully automatic SDS monitored by a latent WoZ operator.

Setup

The second experiment session was located in the same rooms. The SDS was fully automatic although a developer was available to monitor the system and guide the user when he/she had difficulties interacting. Figure 2.17 summarizes the setup. The listening control was left for the system to manage. The user was facing a wireless portable speaker which was similar to a bluetooth hands-free kit for the phone.

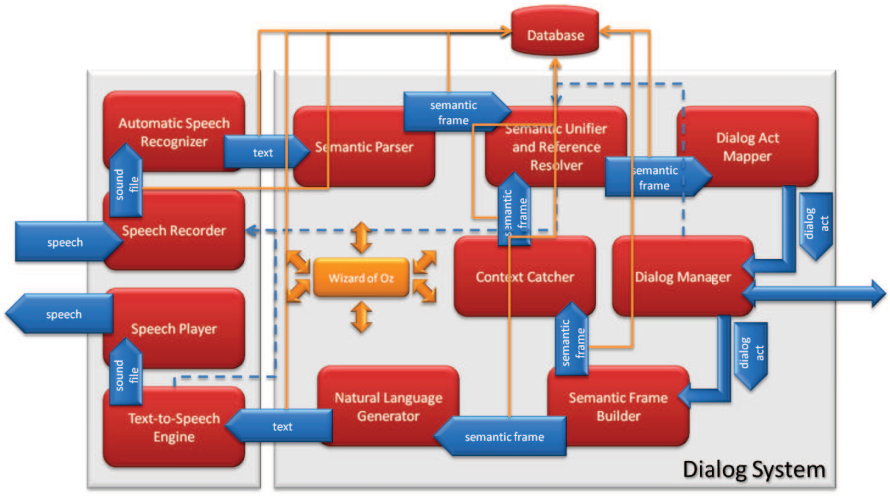


Figure 2.17: Second setup

Users

In total 17 persons participated in the second lab trial.

Table 2.4 below provides an overview on the sample, demographic data and pre-interview results referring to experience with technology; physical constrains that might affect technology interaction; applied coping strategies; doctor visits and the perceived stress level of doctor visits.

Data	Austria	France
Participants	9	8
Age (range)	69.1 (63-76)	77.0 (66-90)
Gender	4 ♀, 5 ♂	7 ♀, 1 ♂
Retiree	9/9	8/8
Experience with touchscreen devices	8/9 have experience with touch-screen devices	5/8 have experience with touch-screen devices
Physical constraints	7/9 fine motor restrictions	5/8 chronic disease
Frequency doctor visits	5/9 less than once a month, 4/9 more frequent	7/8 less than once a month, 1/8 more frequent
Perceived stress level	3/9 a bit stressful or stressful, 6/9 not stressful at all	2/8 a bit stressful or stressful, 6/8 not stressful at all
Reduction of doctor visits requested	5/9 yes, 4/9 no	3/8 yes, 5/8 no

Table 2.6: Second lab trials user sample description and pre-interview results

Collected data and results

A for the first lab trail, the collected data has been aligned and combined so that each entry in the database consists of the sound file's absolute path, the best ASR hypothesis, the semantic parse, the semantic frame out of the NLU sub-system, the system's DA and the CC message (if any). The ASR hypotheses have been manually checked offline.

The database contains 178 dialogs consisting of 603 turns. That is less than 28 minutes of cumulated speech data.

No specific data usage was targeted although the inter-component messages were stored. Some has been used for further training of learning modules.

The data from both sessions provides a corpus for post evaluation of the overall system and its individual components. It contains 253 dialogs (939 turns).

The user feedbacks are reported in Section 3.9.7.

2.8 Conclusion

The main theme of the thesis exposed in the present document is a platform to support the development of SDSs. It is open-source and modular.

Indeed, the research community in the spoken interaction domains lacks tools to experiment with SDSs. Each member generally focuses on a specific topic or component of such systems. The here presented platform allows for the easy and quick substitution of any of the modules it consists of. Moreover, it

does not require an in-depth knowledge of the "other" components to set them up.

This chapter is intended as a primary documentation resource to use the platform, whether the reader is an application designer willing to interface services with an intelligent spoken interaction, or an expert in the domain who may need some support in the implementation of a complete system.

In the next chapters, the components of the ASR, NLU and DM modules are detailed. They have been implemented within the open-source framework described here and so are they open-source as well

Chapter 3

A Step Towards Continuous Listening for Spoken Interaction

3.1 Introduction

The ASR task has evolved quite dramatically in the last decades [75]. Noticeable progresses have been made in the signal parameters extraction, the basic unit selection and the word sequence modeling. Some of these steps have had more impact than others. The speech signal encoding as Mel-Frequency Cepstral Coefficients (MFCCs) [46], the use of HMMs to model the distribution of the descriptors for each acoustic unit [59, 90, 151, 152, 154, 211, 219] and the probabilistic computation of the likelihood of sentences [33, 34, 139, 181, 186, 222] were major steps in this improvement.

Despite these efforts, the technology is not yet reliable enough compared to human performances for the same task, especially in challenging sound environment. Results are tightly related to the recording conditions and the quality and quantity of the training data.

Currently, the market leaders are global companies collecting and learning from huge amounts of proprietary data.

DMs handle user's inputs on a turn-by-turn basis. The unit of action, called the DA, is the minimal effect a user can have on the dialog. Human-human conversations are interleaved interaction turns from participants. Turns can overlap in time and thus it is difficult to identify who has the floor, who is the most important speaker and what is the relevant information to be extracted from his/her/its inputs. In consequence, the first task of a human-machine interaction manager applied to voice is to locate the start and end pointers at the beginning, respectively the end, of a turn, i.e. define the boundaries of a segment of speech so that it is the most relevant in the dialog context.

Automatic classification and recognition methods are ruled by a principle which says that increasing the number of symbols to be identified in the input deteriorates the algorithm performances. Also, the more variations in the training, the worse the specificity of the method. In other words, the more a system fits a cluster of data, the better is its recognition rate. On the other hand, a restricted system does not generalize well, while a generic system can handle more variations, although achieving lower accuracy. The following approach is based on the assumption that the optimal method should use both a specific system and a generic one and combine their outputs to get optimal results.

3.2 Automatic Speech Recognition: An Introduction

3.2.1 Recording Speech

A speech signal, as recorded by a microphone, is an analog time-varying amplitude related to the air pressure on the surface of the sensor. In order for an ASR system to make hypotheses on the orthographic content of a speech signal segment, the waveform has to be digitized, analyzed and converted to a sequence of acoustic parameter vectors [5]. Then the output of this parameterization process can be matched against predefined models of acoustic units.

An analog-to-digital converter samples an analog signal to produce a discrete-valued representation. The configuration of such a process defines the sampling frequency and the output value range.

3.2.2 Parameters Extraction

Over the years, several parameter extraction techniques have been proposed to value the distinctive features of a signal. The currently most widely used method derives sequences of MFCC [46] vectors from a speech waveform.

The signal is first segmented based on overlapping frames then a windowing is performed on every sample. The spectrogram is obtained by the application of the Fast Fourier Transform (FFT), i.e. a computer implementation of the discrete Fourier transform, switching the analysis from the time domain to the frequency domain. Mel-frequency filter banks are applied to the resulting signal representation. Finally, the coefficients are converted back to the time domain via a cosine inverse transformation.

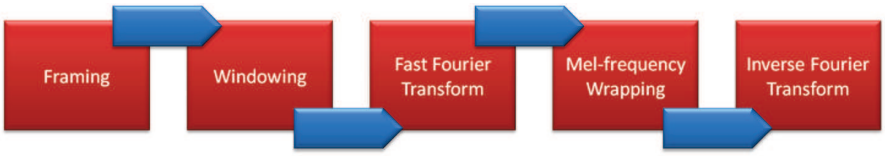


Figure 3.1: Mel-frequency cepstral coefficients computation chain

MFCCs have well-known advantages:

- They give a good discrimination between components
- They are based on the actual human ear perception[1].
- The coefficients are weakly correlated

They are however very sensitive to noise.

3.2.3 Search Graph

An ASR engine searches the best scoring path in a network of sound units. We will consider here that the units of sound are phonemes, i.e. the smallest units of sound. The search graph combines three KSs: the LM, the lexicon and the set of AMs.

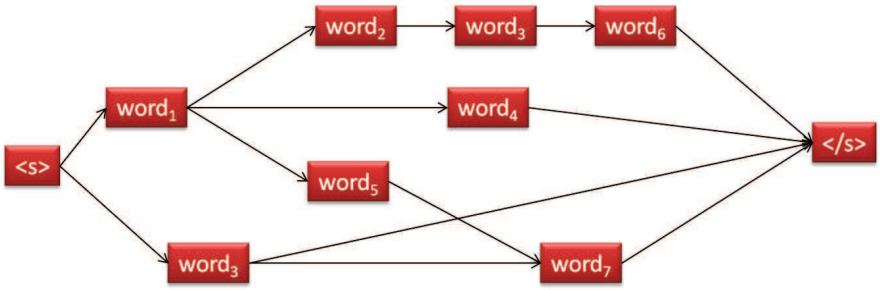


Figure 3.2: Search graph

3.2.4 Language Modeling

An LM defines the authorized combinations of words or symbols for a language. It does so either based on a grammar which strictly defines the available combinations or with some probability functions trained from data, which give the likelihood of a sequence [33, 34, 139, 181, 186, 222].

Context-free grammars

A CFG is formally defined as a tuple $G = (S, N, \Sigma, R)$ where S is the start symbol, N is the set of non-terminal symbols, Σ is the set of terminal symbols also called the alphabet and R is the set of derivation rules.

Starting from S and applying the derivation from R, the process defines all the available sequences of terminal symbols, from the alphabet Σ. The set of those can be finite as well as infinite.

[1]The mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another.

CFGs are well suited for small and grammatically simple languages such as commands or code because they strictly define the symbol sequences. They however can reach high levels of complexity and cannot be inferred from data thus requiring intensive human labor. Moreover, for a NL spoken interaction, an ASR engine should not be configured with them since they do not allow for any deviation from the original static grammar.

Probabilistic context-free grammars (cf. 4.2.3)

PCFGs extend from CFGs with a probabilistic distribution attached to the derivation rules. The sets of symbols Σ and N, including the starting one S are kept. However, the likelihood of a derivation rule is an additional parameter used to discriminate parse trees according to the overall computed likelihood. The value is computed with:

$$P(T) = P(r_1) \times P(r_2) \times ... \times P(r_n)$$

where $P(T)$ is the likelihood of the parse tree T and $P(r_i)$ is the probability associated with the rule r_i. T is built from the application of the rule 1 to n.

Since the derivation rule's likelihood is computed from data, PCFGs encode a small part of the domain knowledge. However, they can grow rapidly and thus are unfit for large interaction scopes.

N-gram model

The current trend for Large Vocabulary Continuous Speech Recognition (LVCSR) language modeling is to use n-grams [33, 34, 139, 181, 186, 222]. An LM based on n-grams is made of the probabilities of words given the $n-1$ previous words.

This likelihood is computed from a domain-specific corpus of text in the target language. The probability of a word w with respect to the $n-1$ previous ones is naively computed with this formula:

$$P(w_n|w_1, w_2, ..., w_{n-1}) = \frac{c(w_1, w_2, ..., w_n)}{c(w_1, w_2, ..., w_{n-1})}$$

where $c(w_1, w_2, ..., w_n)$ is the count of the occurrences of the sequence $w_1, w_2, ..., w_n$ in the training corpus.

The likelihood of a sequence of words $w_1, w_2, ..., w_k$ can then be estimated with:

$$\begin{aligned} P(w_1, w_2, ..., w_k) = & P(w_k|w_1, w_2, ..., w_{k-1}) \\ & \times P(w_{k-1}|w_1, w_2, ..., w_{k-2}) \\ & \times ... \\ & \times P(w_2|w_1) \\ & \times P(w_1) \end{aligned}$$

Preserving the whole history of words to compute the probability of the next one is not efficient and may introduce some defects when the data available for this particular sequence is scarce. To overcome the data scarcity and smoothen the probabilities of an LM, n-gram LMs limit the size of the history to the order n, i.e. only $n-1$ words are kept. The estimation is thus:

$$\begin{aligned} P(w_1, w_2, ..., w_k) = & P(w_k | w_{k-n+1}, w_{k-n+2}, ..., w_{k-1}) \\ & \times P(w_{k-1} | w_{k-n}, w_{k-n+1}, ..., w_{k-2}) \\ & \times ... \\ & \times P(w_2 | w_1) \\ & \times P(w_1) \end{aligned}$$

Injecting the 3-grams probabilities into the search graph, one obtains the graph in Figure 3.3

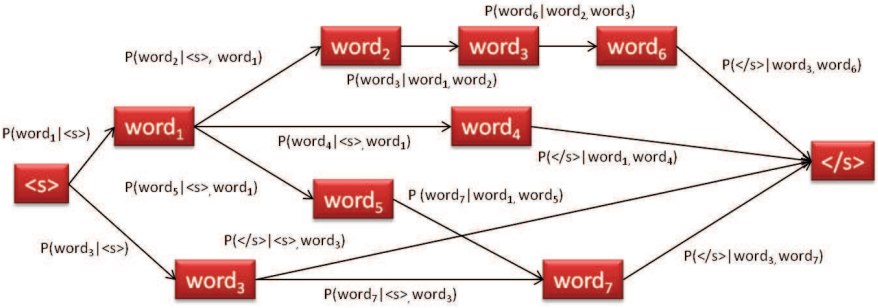


Figure 3.3: 3-gram LM graphic representation

The prediction quality of a model, which is evaluated with the perplexity measure, is highly dependent on the quantity of training data available and on the correlation with the application domain.

3.2.5 Lexicon

A lexicon or pronouncing dictionary provides a decomposition of the words in a language into acoustic units. LVCSR AMs are trained on a phoneme basis, i.e. on the smallest units of sound.

Lexicons are built by experts in phonetics and are specific to a language. They link the words and their distribution in the LM with the AMs.

Figure 3.4 shows the result of applying the lexicon to the previous search graph.

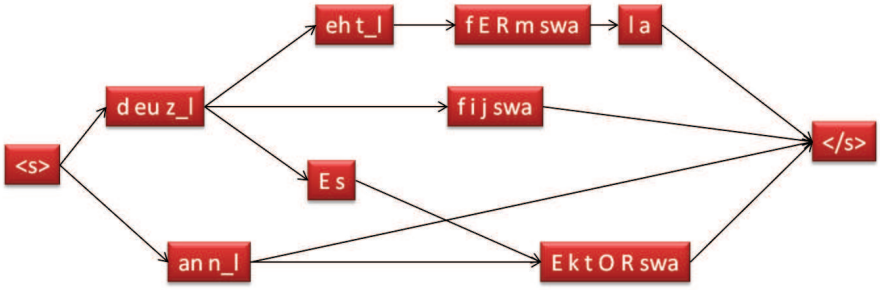


Figure 3.4: Acoustic unit search graph (LM's probabilities have been omitted for more clarity)

3.2.6 Acoustic Modeling

HMM-based AMs associate a generative HMM with each unit of sound [59, 90, 151, 152, 153, 154, 211, 219]. The model estimates the likelihood of phonemes to be contained in a time-variant sequence of observation vectors.

Hidden Markov Model definition

An HMM is a tuple $H = (S, V, A, B, \pi)$. S is the set of states $S_1, S_2, ..., S_N$. V is the vector space. A is the state transition matrix, whose elements are:

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$$

where q_t is the state occupied at time t.
B is the observation symbol probability for each state:

$$b_i(k) = P(V_k | S_i)$$

where V_k is an element of V
π is the initial state distribution:

$$\pi_i = P(q_o = S_i)$$

Based on the model parameters, one can compute the likelihood that such a HMM generates an observation sequence O_1, O_2, O_T where O_i is a symbol from the vocabulary V.

HMM decoding

The purpose of decoding is to compute the likelihood of an observation sequence to be generated by an HMM, i.e. $P(O|A, B, \pi)$ where $O = O_1, O_2, ..., O_T$ is the observation sequence, A is the state transition matrix, B is the observation symbol probability for each state and π is the initial state distribution.

Based on the forward-backward algorithm, the Viterbi procedure efficiently computes the most likely path through an HMM which generates the observation sequence.

For speech recognition, the HMMs generate sequences of descriptors. Each model matches one unit of speech and thus one gets the likelihood of modeled units making up the uttered sequence.

HMM training

In order to compute the parameters A and B for a given HMM, an iterative training procedure takes place. The objective is to maximize the probability of the best path in the HMM generating the training data, i.e. given the set of observation sequences O, tune the parameters for the cumulative probability $P(O|A, B, \pi)$ to be a maximum. This is achieved with Expectation-Maximization algorithms which estimate the parameters so that the models best fit the learning corpus.

Section 4.2.4 presents an additional usage of the HMMs.

In Figure 3.5, the phonemes have been substituted with their HMMs.

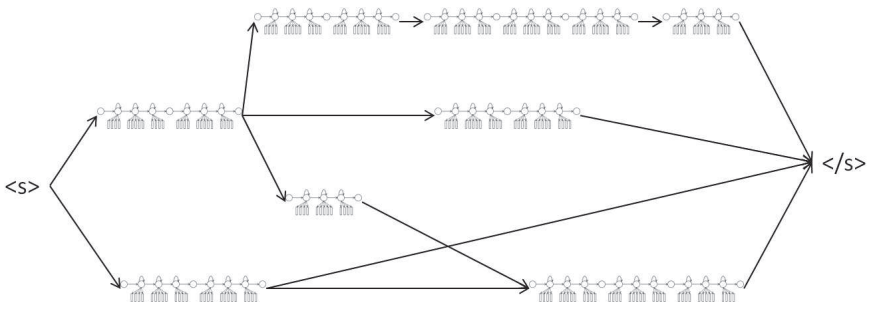


Figure 3.5: HMM search graph (LM's probabilities have been omitted)

Merging the three KSs that are the LM, the AM and the lexicon, this is the graph for the engine to search in.

3.2.7 Software and Tools

HTK

The Hidden Markov model ToolKit (HTK) [184, 219] is an open-source set of publicly available tools. It was originally developed by the Department of Engineering at the University of Cambridge.

The HTK is used to manipulate HMMs, which can be used for speech synthesis and recognition, handwriting recognition, DNA sequencing and many other applications. It consists of modules, libraries and tools developed in the C programming language. The software supports both continuous mixture Gaussian

distributions and discrete ones, which allows to create complex systems based on HMMs.

Julius

Adapted from the Julius website's homepage (`http://julius.sourceforge.jp`)

Julius [104, 105] is a high-performance, two-pass LVCSR decoder software for speech-related researchers and developers. Based on word n-gram and context-dependent HMM, it can perform almost real-time decoding on most current PCs in 60k word dictation tasks. Major search techniques are fully incorporated such as tree lexicon, n-gram factoring, cross-word context dependency handling, enveloped beam search, Gaussian pruning, Gaussian selection, etc. Besides search efficiency, it is also modularised carefully to be independent from model structures, and various HMM types are supported such as shared-state triphones and tied-mixture models, with any number of mixtures, states, or phones. Standard formats are adopted to cope with other free modelling toolkits such as the HTK, the CMU-Cambridge Statistical Language Modeling toolkit, etc.

The main platform is Linux and other Unix workstations, but it also works on Windows. The most recent version is developed for Linux and Windows and also has a Microsoft SAPI version.

Julius is distributed with an open licence together with source codes.

Julius has been developed as a research software for Japanese LVCSR since 1997. The work was continued under the IPA Japanese dictation toolkit project (1997-2000), the Continuous Speech Recognition Consortium, Japan (CSRC) (2000-2003) and currently in the Interactive Speech Technology Consortium (ISTC).

Sphinx

Sphinx-4 [198] is a state-of-the-art speech recognition system written entirely in the Java programming language. It was created via a joint collaboration between the Sphinx group at CMU, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL), and Hewlett Packard (HP), with contributions from the University of California at Santa Cruz (UCSC) and the Massachusetts Institute of Technology (MIT).

Sphinx-4 started out as a port of Sphinx-3 to the Java programming language, but evolved into a recognizer designed to be much more flexible than Sphinx-3, thus becoming an excellent platform for speech research.

Sphinx includes features such as the live mode and batch mode speech recognizers, capable of recognizing discrete and continuous speech and the generalized pluggable LM architecture, which includes pluggable LM support for ASCII and binary versions of unigram, bigram, trigram, Java Speech API Grammar Format (JSGF), and ARPA-format FST grammars.

3.3 The most common listening method

Figure 3.6 presents a schematic of the most commonly used listening method for the current SDSs.

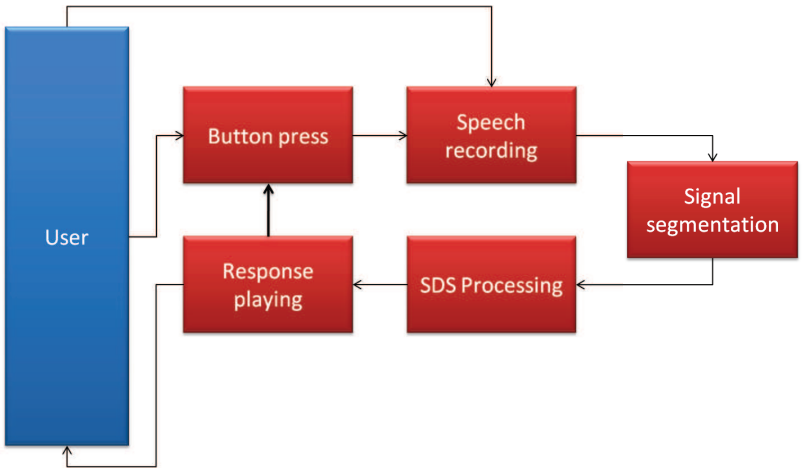


Figure 3.6: Most common listening method for SDSs

Here is how it works.

1. The user signal its intent to address the system via the press of a button.
2. The start-of-user-segment button is disabled while the recording is triggered.
3. Depending on the method that is used, the system or the user marks the end of the user speech segment.
4. The resulting speech segment is processed by the SDS.
5. Once the SDS has produced a response, it is played back to the user.
6. After the system segment is done the button is re-enabled, thus the system gets back to its initial state.

In order to achieve a truly human-like listening, starting from this generic method, there are four problematics to be tackled. The sequence of steps is the following:

- Press-to-speak-button removal: the system gets a continuous audio streaming
- Robust segmentation: the system gets segments of audio signal

- Noise filtering: the system gets only speech segments
- Intelligent attention selection: the system gets only speech segments that are intended to the interaction with the interface.

In this chapter, the contributions of that thesis toward achieving a truly continuous listening for the spoken interaction are presented.

3.4 CompanionAble Project Setup and Task

Within the CompanionAble consortium, the task at Institut Mines-Telecom was to add a speech recognition ability to an overall ambient intelligence system.

To capture sounds, a wireless omni-directional CMT microphone had been mounted on the "head" of a mobile companion robot [69]. The Speech/Sound Analysis (SSA) module got the input signal from a wireless sound card provided by the Austrian audio hardware producer AKG. The analog signal was sampled at 16 kHz on 16 bits. A single orthographic hypothesis over the content of a spoken utterance was transmitted to the central controller to be consumed by the dialog management system implemented in the robot's software.

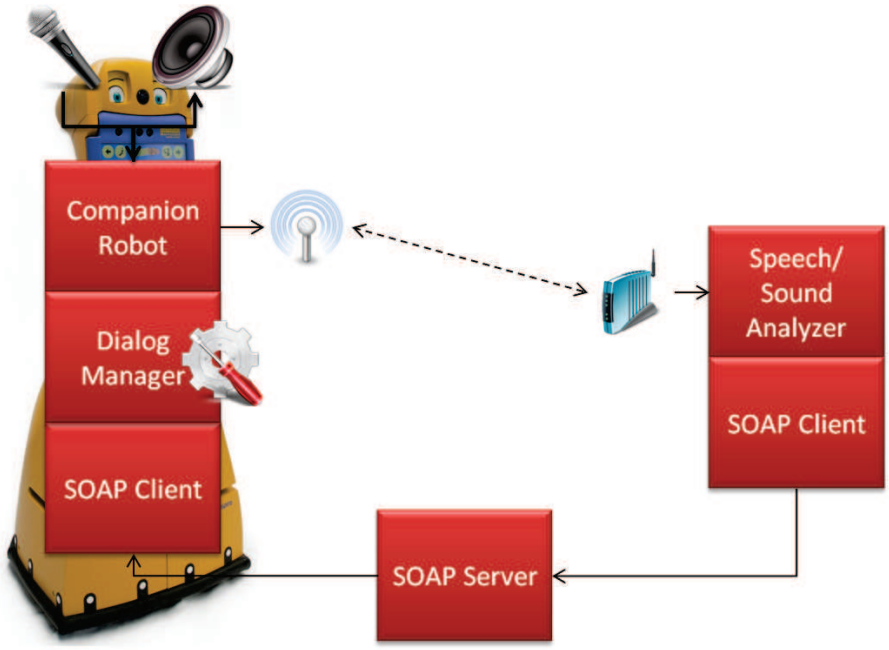


Figure 3.7: CompanionAble's architecture sample

The main trial sites were in Eindhoven in the Netherlands and Gits in Belgium, which are Dutch-speaking and Flemish-speaking locations, making Dutch

the trial language. This led to further developments and tests with this language. Notice however that speech recognizers for English, French and Spanish were built as well.

3.5 Speech Recognition Issues

A first prototype was built to experiment with the system. This enabled us to point out some weak points of the system.

3.5.1 Acoustic Mismatch

The project's target user group was dependent European senior citizens living at home. The speech characteristics of this age category are different from those of the younger population due to age-related physical and cognitive transformations [43, 66].

Acoustic phonemic HMMs trained on readily available transcribed-speech databases, such as ESTER/ETAPE [62] (TV and radio broadcasts) for French or the Corpus Gesproken Nederlands (CGN) for Dutch, do not fit well the speech descriptor sequences produced by such users.

Consequently, a better matching has to be obtained to get an acceptable recognition rate. This can be achieved by either retraining the HMMs from scratch with elderly speakers recordings, or by adapting them with less data.

3.5.2 Distant Speaker

As previously mentioned, the sound-capturing device uses a head-mounted CMT microphone on top of a mobile companion robot. The machine could move freely into the user's house and the SSA system was always "listening". The user may address the robot anytime and from anywhere within the broad omni-directional recording range of the microphone. The distance to the speaker had a major impact on the overall ASR performance due to the increase or decrease of the signal's amplitude [121].

3.5.3 Echo

The house spatial configuration induced that the user speech may be reverberated depending on the location of the user and the companion robot. The sound waves originating from the speaker's mouth bounced off the floor, the walls, the ceiling and large objects.

The captured signal was disrupted by other weaker echoed segments. The added delay and noisy speech had a negative effect on the ASR performances.

Moreover, since the CMT microphone was mobile, pre-processing algorithms to remove the echo were blind, i.e. they could not rely on accurate models of the environment or assumptions about the recording conditions.

3.5.4 Uncontrolled Background Noise

Issues related to the spatial location of the robot are not the only causes of disturbances. Noise sources are plenty in a house. Media such as radios or TV sets, electronic appliances such as washing machines, microwave ovens or vacuum cleaners produce some when turned on. The noise from key shuffling, doors opening/closing, running water, steps, coughs, glass breaking or rain on windows has to be filtered out to analyze speech-only signals.

Among the methods that have been applied to remove these signal parasites, one aims at modeling the noise and then discard it [199]. Another one puts sensors near known sources of noise and removes the recorded signals from the interesting ones [102, 103].

3.5.5 Controlled Background Noise

The noise produced by the robot (audio player, driving motors) is a factor that can be somehow controlled, i.e. removed from the signal. Indeed, the system "knows" both the content and the time of emission of these robot-originating sounds. It then relates to the second method from the previous subsection to cancel those recorded segments out.

3.5.6 Single Input Channel

The system employs a single microphone to record the environment through a single channel in a single unknown mobile location of the house. Speaker tracking methods or microphone array-based source separation algorithms can not be applied.

3.5.7 System Attention

Even if an ideal clean environment is obtained in which only the speech signals are analyzed, there is still the issue of getting the attention of the companion robot. The system must collect clues as to know whether a detected speaker is addressing the ambient intelligence, is on the phone, is listening to the TV, to the radio or even talking to guests or caregivers. An unimodal, speech-only triggering mechanism has to be integrated.

3.6 Primary Continuous Listening System

3.6.1 Architecture

Within the SSA system, the ambient sound signal is continuously analyzed by two parallel modules: one that is able to detect and classify sound events and another one which does the speech recognition part. Figure 3.8 shows the parallel processing of these components whose communication is achieved through the TCP/IP protocol. The speech recognition output goes through a

similarity test, then it is filtered with the sound recognition system results in order to reject hypothesis made from the analysis of non-verbal signal segments. The recognized commands are sent to the CompanionAble home server using the Simple Object Access Protocol (SOAP).

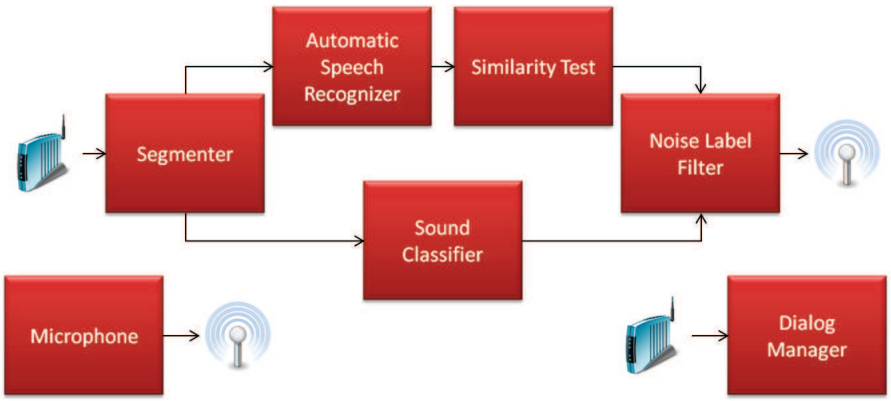


Figure 3.8: SSA system architecture

3.6.2 Signal Segmentation

The first task to carry out before attempting to recognize speech in a continuous signal is to segment it. This process is based on the only 2 dimensions available at that stage, which are the amplitude and the time.

The segmenter in that system places end markers according to a time delay and a level threshold. When N successive windows, which have an energy level below the threshold, are detected, the system stops the recording, sends the last segment to be analyzed and starts a new one. N actually derives from a measure of time expressed in milliseconds:

$$N = F \times L$$

where F is the sampling rate and L is the length of the time delay (in seconds).

3.6.3 Sound Classification

This module was conceived and implemented by the École Supérieure d'Ingénieurs en Informatique et Génie des Télécommunications

The sound classification is a two-steps process [84]. It is the combination of a detection module based on wavelet transformation and a hierarchical labeling system (noise/speech and sound classification) based on Gaussian Mixture Model (GMM). The sound classes used for the CompanionAble trials were

trained using sound recordings collected with CMT microphone in the experimentation house. The system integrates five sound classes: fall of an object, door bell, key shuffling, cough and claps. The sound classification is a preprocessing of the speech analysis, filtering out signal segments labeled as noise.

To reduce the processing time, the two modules are running in parallel, thus synchronization flags are needed. The sound module memorizes the three last noise/speech segment labels associated with timestamps. Then the decision of forwarding or rejecting the speech recognition hypothesis within the server is effective on output pairs with matching timestamps.

The noise classification algorithm was initially evaluated on pre-recorded data. The computed rate was approx. 80% valid. The speech/noise decision was tested in the actual trial environment with an accuracy rate of 95%.

3.6.4 Speech Recognition

The end-user difficulty (even inability) to interact with a computing system through traditional menu-based interfaces and touch screens motivates the deployment of vocal user interfaces. Cognitive or motor-skills deficiencies could drastically lower the system usability if it is not equipped with a distant speech recognition ability.

Labs for practical experiments in the CompanionAble project were based in the Netherlands ands in Flemish-speaking Belgium, thus the final HCI language was Dutch.

Julius, developed at the Kawahara lab of Kyoto University [104, 105], was our choice as a Speech recognition engine. As previously discussed, it is able to process large-vocabulary search in real time, running a 2-pass algorithm. The configuration of the engine consists of 3 KSs: an n-gram LM, a set of acoustic HMMs and a lexicon. In addition, Julius parameterizes digital speech segments on the fly according to an HTK-compliant parameter set. The engine can concurrently process identical audio segments with several instances. All those features made it the appropriate automatic speech recognizer for the CompanionAble tasks.

The phonemic HMMs were trained on the CGN which contains 800 hours of transcribed Dutch speech consisting of nearly 9 Million words. This is the largest corpus for contemporary Dutch. Files are single speaker and multiple speakers recordings of prompted or spontaneous speech.

3.6.5 Language Models Interpolation

The first version of the speech recognition module was based on a single n-gram LM trained on a sub-part of the CGN corpus. The AMs were adapted to fit the voice characteristics of the users using an Maximum Likelihood Linear Regression (MLLR) method [70].

This system produced too much false positives, i.e. unintended commands, when put to practical tests and the recognition accuracy was too low. In order

to improve both the recognition and rejection rates, the approach, described next, was implemented.

Frame-specific models

The dialog is designed based on a frame paradigm. Frames contain non-overlapping sub-dialogs. Transitions between states are triggered by the robot's internal states/variables combined with the user inputs (vocal commands, buttons or/and sensor readings). A frame is enabled, i.e. available, when at least one of its activation conditions is fulfilled; these are the same variable types as the intra frames ones. Thus one can build a dialog hierarchy: the root frame, which is initially enabled, contains all the activation events to enable the sub frames, and intra-frame terminal states allow the sub frames to return the control back to the main frame.

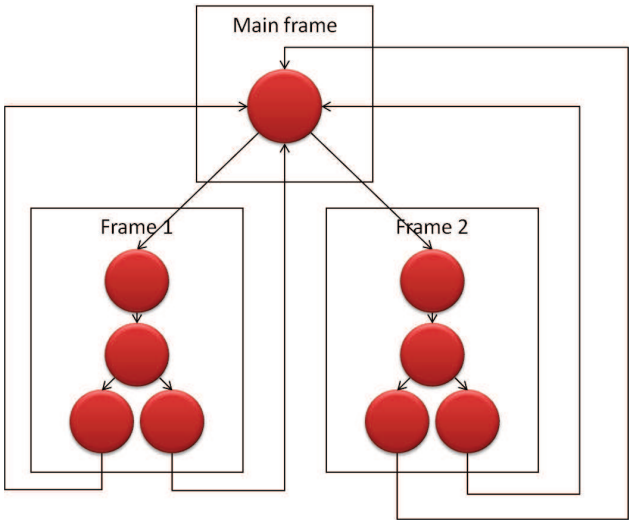


Figure 3.9: Dialog hierarchy illustration

The dialog sub-frames have been clustered into eight classes. Each class lists all the vocal commands which are allowed and can be interpreted by the compound frames. A LM is built from those lists. A 9th LM is trained on the activation commands and is associated with the main frame. Since the speech recognition module does not receive feedbacks about the current state of the dialog, nine parallel instances of the recognition engine are decoding the sound stream and deliver transcriptions.

This LM selection process improves the recognition rates for the application commands but does not solve the rejection issues for out-of application sentences.

Models weights

Section 3.6.6 will discuss the proposed similarity test. It compares the recognition results from each of the nine LM-specific engines with those from a general recognizer. The general recognizer LM is learned from the CGN. Moreover, one has to make sure that the general ASR engine recognizes the sequences of words contained in the specific LMs. One needs to add the whole set of commands in the training corpus of such a general model. We introduced a weight for these additional sentences, which has been experimentally defined to be 1000: the commands were added 1000 times in the training corpus.

The test is not applied to a single general-engine hypothesis. We found that it is better to use the n-best ones as it improves the recognition rate:

- One hypothesis is produced by each specific decoder based on the size of their LM.
- Several hypothesis (3 in our application) are produced by the general decoder and then fed to the similarity test

3.6.6 Similarity Test

The similarity measure between two recognizer hypotheses is an extended Levenshtein distance. This is the total count of needed operations (substitutions, deletions, insertions) to transform a sequence of words into another one. It is furthermore normalized by the count of words in the sequences.

$$NormLev(Ref, Hyp) = \frac{Lev(Ref, Hyp)}{\frac{word\_count(Ref)+word\_count(Hyp)}{2}}$$

For instance:

$$NormLev(\text{“I want to send a message”, “can you send a message”}) = \frac{3}{\frac{6+5}{2}} = 0.55$$

Depending on the relative value of this distance, given a threshold, the hypothesis recognized by a speech recognition instance set with a specific LM is validated or discarded. This test is useful to:

- Confirm a good recognition hypothesis: a well recognized command, according to both the general decoder and the specific decoder is validated. The specific-decoder hypothesis is sent
- Reject a wrong hypothesis: a command recognized only by the general decoder is rejected
- Correct a partially correct hypothesis: a command recognized by a specific decoder while the general decoder outputs a close match, is corrected. The specific-instance hypothesis is sent

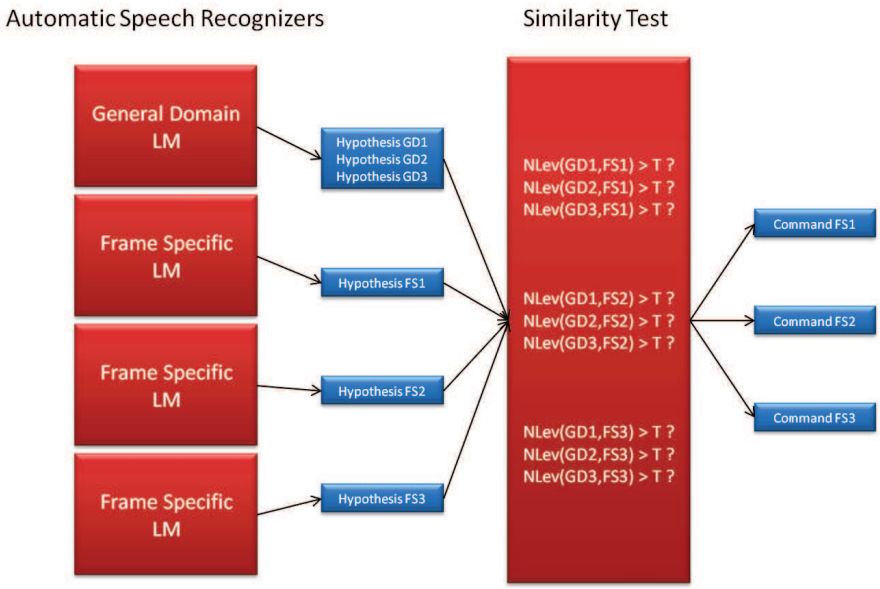


Figure 3.10: Multiple-instance ASR decoding and similarity test

3.6.7 Noise-labeled Segments filter

A LVCSR recognition engine such as Julius searches the best match between the acoustic observation vectors and a sequence of words. One may add a garbage model, which would be the default match to unknown observation sequences or sub-sequences. In our system however, every input is matched with a sequence of words. Thus, noise is processed as if it was speech and a word sequence hypothesis is returned. The speech/noise labeling prevents this to be propagated to the DM by discarding speech recognition results that occurred while the segment was classified as noise.

3.6.8 Attention Level

The DM designers offered to lower the false-positive rate with a dynamic trigger-word detection mechanism. A detection of the keyword in the audio stream increases the attention level of the DM, which otherwise steadily decreases over time. When the level is positive, it triggers the analysis of the recognition results.

The attention level is initially null, the speech recognition engine permanently processes the audio stream and produces text hypotheses. As long as the trigger word can not be extracted from those text segments, the transcriptions are ignored by the DM. As soon as the attention value is greater than zero, a dialog starts and the manager processes the received transcribed commands.

While the dialog is sustained between the user and the system either by repeating the trigger word or by moving through the dialog flow, the attention level increases. Silences and out-of-scope utterances (rejected by the SSA system), on the other hand, lower the level. If the floor value, i.e. zero, is reached, the analysis of the inputs stops.

The selection of the attention word is important for the stability and reliability of such a mechanism. It has to be easy for users to remember but should be distinct from any other word that may be detected in the background environment or in casual conversations.

3.6.9 Acoustic Adaptation

Acoustic adaptation methods have been studied at the earliest stage of the project. Two adaptation methods were compared: the Maximum A Posteriori (MAP) [63, 110] and the MLLR [27, 70]. A LM has been trained on a corpus of 57500 sentences derived from practical experiments and paraphrasing. The speaker was the same for the whole study. She had been previously recorded and the audio files were played through a loudspeaker.

Only 10 phonetically balanced sentences were used to train the system, the MLLR adaptation was the technique for getting the best results. Without adaptation, 60% of the transcriptions are correct. This rate reached 70% with MAP adaptation and 73% with MLLR adaptation.

Users went through a preliminary MLLR adaptation round before they used the system.

3.7 Evaluation

3.7.1 Expected Improvement Axes

The improvement over a basic ASR system that were expected to be brought by this setup are two-fold.

The first axis aims at increasing the recognition rate of the system for in-domain utterances. The reliability is itself two-fold. It includes getting a good rate while attempting to transcribe in-application commands and automatically reject out-of-scope utterances to avoid misleading the DM.

A second objective is to estimate the degree with which a segment has been intended to the SDS. Since the listening is continuous, the system needs to select the speech segments that are addressed to the machine and discard all the others.

3.7.2 Improving the ASR Reliability

The reliability is optimized thanks to the combined action of the speaker adaptation, the threaded speech recognition, the sound classification and the similarity test. This last mechanism allows for:

- validating in-application utterances
- discarding out-of-scope hypotheses
- correcting partially recognized sentences

3.7.3 Detecting the Intended Segments

The attention level, which gives the estimate of the intention of the user to interact with the machine, is computed from the detection of a keyword, the dialog progress measure, the threaded speech recognition processes and the similarity test.

3.7.4 Evaluation Corpus

A test corpus with 5 speakers has been recorded in the demonstration building. Each of them uttered 58 sentences: 10 phonetically-balanced sentences for acoustic adaptation, 20 in-application commands, 22 out-of -application commands and 6 partial commands. The partial commands were obtained by deleting one or more words in in-application commands.

3.7.5 Noise-free Evaluation

Objectives

The first phase was intended to set the value of the commands' weight in the general model and the number of hypothesis from the general recognition process used in the similarity test.

Setup

The evaluation setup is shown in Figure 3.11. Audio test files are played through a loudspeaker and recorded by a CMT microphone. A second loudspeaker was used in the second phase of the experiment to simulate noisy conditions. The loudspeakers were placed on top of each other. The sound level was set to be about 60 dBA which is the level of an average speaker standing one meter away from the recording device.

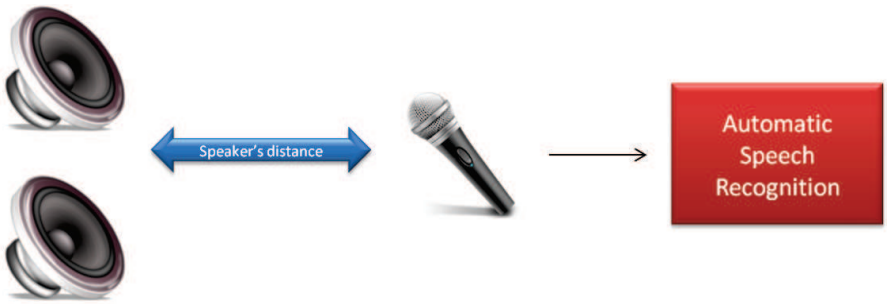


Figure 3.11: Evaluation setup

System configurations

Several system configurations have been experimented. The most relevant ones for the evaluation are presented in the following tables. Here is a quick description of each configuration.

- Baseline + adaptation: the baseline system is made from the raw Julius speech recognizer whose acoustic models have been adapted to the test speaker's voice characteristics with the MLLR method using 10 phonetically balanced utterances. Only the best hypothesis is used to compute the evaluation measure.
- Baseline + adaptation + similarity test (specific LMs' weight: 1; general decoder hypothesis: 1): in addition to the previous system baseline setup, the similarity test (cf. 3.6.6) is applied to the speech recognizers' outputs. The specific LM's weight is the weight of the in-application commands added to the general-domain language model training corpus. Only the best hypothesis from the general decoder is input to the similarity test.
- Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 1): the difference between this configuration and the previous one is that, in that case, the commands are added a thousand times to the general-domain language model training corpus. Hence, the specific LMs' weight is 1000.
- Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 3): in the last configuration, the similarity test gets 3 hypotheses from the general decoder, the 3 best ones. As before, the in-application commands' weight is 1000.

Metrics

Two metrics are presented, the sentence error rate:

$$sentence\ error\ rate = 100 \times \frac{count\ of\ badly\ recognized\ sentences}{count\ of\ sentences}$$

And the sentence false-positive rate:

$$sentence\ false\text{-}positive\ rate = 100 \times \frac{wrongly\ validated\ sentences}{count\ of\ sentences}$$

Results

System	Sentence error rate	Sentence false-positive rate
Baseline + adaptation	85% (±7%)	-
Baseline + adaptation + similarity test (specific LMs' weight: 1; general decoder hypothesis: 1)	80% (±8%)	10% (±6%)
Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 1)	45% (±10%)	0%
Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 3)	15% (±7%)	0%

Table 3.1: Sentence error rate and false-positive rate for in-application commands

System	Sentence error rate	Sentence false-positive rate
Baseline + adaptation	91% (±5%)	-
Baseline + adaptation + similarity test (specific LMs' weight: 1; general decoder hypothesis: 1)	100%	0%
Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 1)	100%	0%
Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 3)	100%	0%

Table 3.2: Sentence error rate and false-positive rate for out-of-application commands

System	Sentence error rate	Sentence false-positive rate
Baseline + adaptation	83% (±13%)	-
Baseline + adaptation + similarity test (specific LMs' weight: 1; general decoder hypothesis: 1)	67% (±17%)	0%
Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 1)	33% (±17%)	0%
Baseline + adaptation + similarity test (specific LMs' weight: 1000; general decoder hypothesis: 3)	33% (±17%)	0%

Table 3.3: Sentence error rate and false-positive rate for partial commands

Only 15% of the in-application vocal commands were recognized by the baseline engine whose LM had been trained on the CGN. The featured similarity test improved these results by up to 20% but one could notice the increase of the false-positive rate. The best system could recognize commands with 85% accuracy and never gave false-positives. Every out-of-application sentence was rejected by the system. Partial commands were most of the time validated.

3.7.6 Evaluation in noisy conditions

Objective and setup

In a second phase of the evaluation process, the noise robustness was tested. A second loudspeaker placed on top of the first one played various ambient noises.

Results

Noise type	Sentence error rate	Sentence false-positive rate
Washing machine	26% (±9%)	11% (±6%)
Dutch speaker	47% (±10%)	11% (±6%)
Music	53% (±10%)	5% (±6%)
Crowd	58% (±10%)	11% (±6%)

Table 3.4: Sentence error rate and sentence false-positive rate for in-application commands

Washing machine	100%	0%
Dutch speaker	100%	0%
Music	100%	0%
Crowd	100%	4% (±4%)

Table 3.5: Sentence error rate and sentence false-positive rate for out-of-application commands

Washing machine	60% (±17%)	0%
Dutch speaker	40% (±17%)	0%
Music	80% (±14%)	0%
Crowd	40% (±17%)	0%

Table 3.6: Sentence error rate and sentence false-positive rate for partial commands

As expected, the performances dropped. The recognition rate was lower as well as the rejection rate.

3.8 Another System for Hand-held Devices

3.8.1 Introduction

The previous method may not be optimal for other setups. Such became apparent as later on a system based on spoken interaction posed additional, so far unseen challenges (and offered alternative solutions).

3.8.2 Listening Context

The vAssist project aims at providing a set of services targeting elderly people mediated by a vocal interaction. The endpoints for users to access the system are hand-held devices such as smartphones, tablets, etc.

The vAssist architecture is based on a remote-server implementation of an SDS, which itself is reliant on other distant services. The user interface is a graphical application which enables users to place calls to the SDS and receive/send asynchronous messages from/to a data bus.

Some of the issues mentioned earlier are not relevant to this setup. These include the speaker's distance, the echo, and the attention of the system. Indeed, we made the assumption that a user would be, when connected to the server-based system, close to the microphone and involved in the interaction. The background noise is also assumed to be weak given the microphone limited range of the portable devices.

A user, however is only able to open and close the communication channel to the SDS, he/she has no means of controlling the listening. In calls such is left entirely to the system.

3.8.3 Architecture

The listening system, i.e. the set of components making the decision whether a transcription hypothesis should be validated or rejected spans from the recorder/segmenter to the edge of the DM. The modules are those within the black border in Figure 3.12

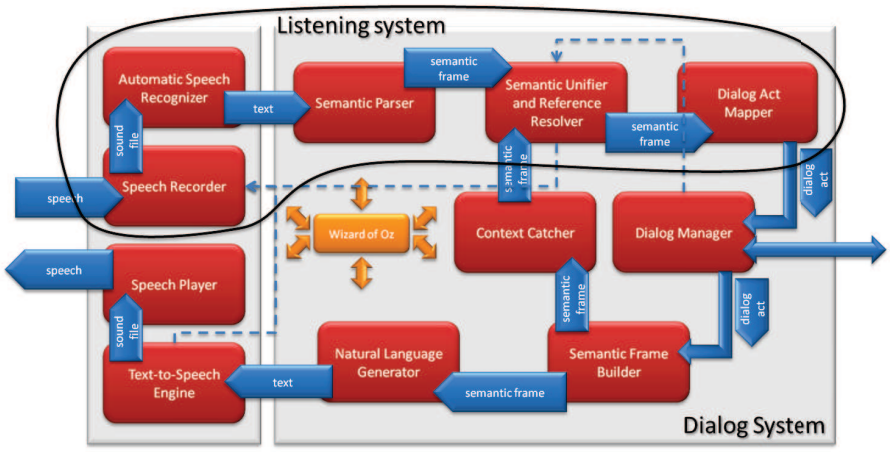


Figure 3.12: Listening system architecture

The speech recorder is responsible for recording and segmenting the signal before the ASR service makes hypotheses about the content of each segment. The SP, the SURR and the DAM define the core of the NLU system (chapter 4).

These components implement some functionalities to estimate the relevance of a segment and thus reject it or process it. DAs are produced out of this inner system when relevant to the current dialog context.

3.8.4 Signal Segmentation

The way the segmenter splits the signal is identical to the method detailed earlier, except for the addition of a terminal binary decision.

The module, when recording, measures the signal amplitude. For a segment to be started, this value has to exceed the threshold defined in dB. In order to end a segment, the algorithm analyses the signal and cuts it when it can spot a period of S seconds during which the amplitude of all the frames of the signal are below the pause level.

Moreover, a minimum duration parameter has been defined. Any segment whose length is less than it is rejected. The idea of this mechanism originates from the fact that, when connecting to the SDS with a smartphone, we observed that some users, including us, manipulate the client device while interacting. It produces noises which have the property of being short bursts of sound that

are processed by the system. Implementing this minimum duration condition proved to be an efficient way of "cleaning" the captured signal.

The parameters of the segmentation are dynamic, i.e., they can be changed when the segmenter is not recording. This gives the possibility to update them in real time and learn them from the on-going interaction. However, the here presented work does not present results of such experiments.

3.8.5 Confidence Scoring

The second control mechanism for filtering the segments is in the ASR module. This service makes hypotheses about the content of segments. To achieve that, the signal is parameterized and the sequence of descriptor vectors are matched against AMs. The confidence score reflects how similar the segment to be recognized is to a modeled word sequence [96, 97].

The ASR engine returns an n-best list of transcriptions ranked according to the computed score. Hypotheses whose value is below the threshold are discarded. They are, however, not completely ignored but rather converted to a non understanding SF, which is passed on to the next module in the SDS.

3.8.6 Semantic Appropriateness

In a dialog between a human and a machine, the purpose of the NLU component is to associate a DA (unit of dialog move) to the user input.

A continuously listening interface attempts to interpret every recorded speech segment. However, the conversion to a DA may not be obtained, depending on the user input and/or the current dialog state.

The first transformation applied to the transcribed utterance is the semantic parsing. This is a partial one, i.e. only parts of the sentence are labeled with semantic concepts. The first control mechanism to remove an incorrect utterance is then to reject any input from which no SF can be built. Actually, the system produces a default empty SF for such cases. This to enable error recovery mechanisms.

Then the SURR processes the resulting SF to contextualize it. Here again, if no suitable frame can be obtained from the algorithm, an error message is passed on to the next component.

The last component a transcribed hypothesis goes through before entering the DM is the DAM. This module, tightly joined with the DM, has access to the dialog state maintained in the SDS core. From that knowledge, the DAM defines a set of DAs that, in the current situation, have an effect on the dialog evolution. Matching an SF with a DA is straightforward when the latter is available. Whenever an SF can not be converted to a DA included in the DAM-made set, it triggers a non understanding error. It means that either the system misinterpreted the user utterance or the user request is not available at that point.

Thus, the semantic appropriateness of the transcribed speech is tested, not only at the acoustic and linguistic level but also through semantic analysis.

These checking mechanisms reinforce the robustness of the overall input processing system.

3.8.7 Error Recovery

Recovering from errors implies several tasks: detecting the errors, tracking the error source, automatically resolving the mistakes or, in some cases, signaling the error and guiding the user to recovery [80].

We have seen previously that the input processing pipeline implements some milestone checks to reject a signal segment at several abstraction levels (acoustic, linguistic, semantic, dialog). It has also been mentioned that the erroneous inputs were not completely rejected but actually generated SFs whose goal is "non understanding".

The SURR keeps track of these SFs. When three successive non understandings occur, the two first ones are ignored, i.e.the SURR stops the processing and reactivates the listening. This is to prevent the system to produce error messages too often when recording out-of-scope interaction turns, which may not be intended to it. In that, ignoring two errors out of three successive ones appears less annoying to the user. Two is simply an arbitrary number and could easily be changed.

Whenever a non understanding frame enters the DM, the triggered dialog contains one turn: it signals the user that an error occurred and tries to give some clues about how to recover from it. Clues are the best hypothesis of the ASR, the dialog state in human-readable form and what is available and expected in that state. The system's turns include them progressively. They:

1. signal that a non-understanding occurred: "I didn't get that"
2. signal that a non-understanding occurred including the best ASR hypothesis: "I heard <best hypothesis> but I can not understand it in this context"
3. signal that a non-understanding occurred and propose some utterance templates to the user: "I cannot understand that. Try to say <template 1> or <template 2>"

3.8.8 Multiple Hypothesis Testing

Often, what we mean to say can be transcribed in different ways. The surrounding context (sentences or words) can help an ASR engine to select the right word sequence. This statistical information is encoded within the LMs. Moreover, in a dialog, there is an iteratively built shared interaction context, the user turn may be very short and thus not include much clues about the correct transcriptions to produce.

As an example, in French, "10 heures" which means "10 hours" or "10 o'clock" is pronounced the same as "Deezer", a French music streaming website. For instance, a user utterance containing only "10 heures", in response to a

starting point request, produces a non-understaning SF when transcribed as "Deezer". However, looking at the n-best hypotheses from the ASR, one can see that the correct transcription is ranked second or third.

The platform processes up to seven hypotheses from the same speech segment. If the best one can not be associated with a DA in the current context, the second one in the list is tried, and then the third and the fourth one until the last hypothesis of the initial set. If the system gets to the end of the list without finding an appropriate DA, an error frame is produced.

"10 heures" is now always understood (when in the right context), whatever the first hypothesis is.

3.9 Evaluation

Similarly to the first setup, this second system for hand-held devices have been experimented with real users.

3.9.1 Introduction

Two listening methods have been experimented with, along the course of the vAssist project.

The first experience session took place in Fall 2013 in Paris, France and in Vienna, Austria, early in the development process. The goal was to estimate the usability and validate the functionalities of the system in a user-centered manner. The setup included a WoZ which replaced the components which were not developed at the time.

The second session started in Spring 2014 based on a fully automatic system. The goal was to validate the final version of the system in the lab before offering it to actual users for an extended period of time (several months) in an uncontrolled environment.

3.9.2 First Setup

Figure 3.13 shows the setup for the first experimental session.

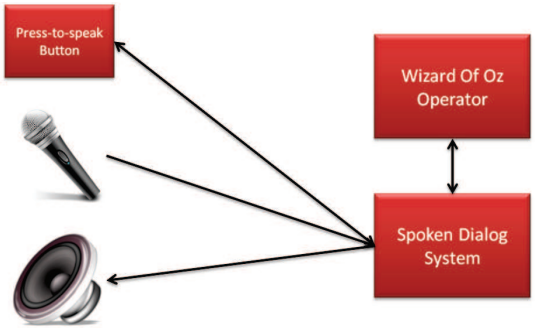


Figure 3.13: First setup

The location was the experience room of the lab. The noise was limited to the one produced by people inside the room. The user was facing the laptop displaying the press-to-speak button and integrating the microphone and two speakers. The facilitator was seated next to the experimenter.

The WoZ operator, who was outside the room, received the user's utterances as transcribed by the ASR service and created the SF expressing the intent on the fly. This data structure was sent back to the automatic system which contextualized SFs, managed the dialog, generated and synthesized spoken responses.

The recommendation to the user was to click on the press-to-speak button before uttering commands. The button label then became "Listening..." until the signal amplitude fell below the threshold for long enough to be detected as a silence. Past this delay, the listening was stopped and the button showed "Processing...", then the system played a response and the button was made available again.

3.9.3 Observations

The main issue with this setup was the required button press. Elderly users often forgot to activate the listening, which was harmless for the system, but quite frustrating for the human speaker. However, they quickly got used to trigger the recording after a few turns.

The button was "locked" while the system was processing the inputs and playing replies. This added to the frustration of the user and to the feeling that the system was slow. Actually, most of the speed reduction of the system came from the WoZ latency and the Internet connection quality.

Intra-utterance pauses were also an issue in this setup. Users were thinking about what they were saying while speaking, which increased the silence between words. They were detected as the end of utterances and thus led to non understandings or partial ones.

3.9.4 User Feedback

In post-experiment interviews, users expressed their satisfaction with the listening control after they remembered to press the button.

The information displayed on the button was not always noticed and users complained about how non-informative the SDS was about its internal status.

3.9.5 Second Setup

The second experiment session was located in the same room. The SDS was fully automatic although a developer was available to monitor it and guide the user when he/she had difficulties interacting. Figure 3.14 summarizes the setup. The listening control was left for the system to manage. The user was facing a wireless portable speaker, which was similar to a bluetooth hands-free kit for phones.

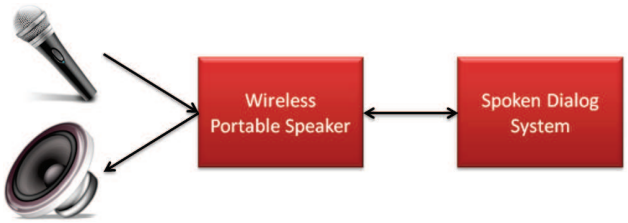


Figure 3.14: Second setup

3.9.6 Observations

In comparison to the strict sequencing of the dialog in the first experiments, the second setup showed more naturalness in the turn taking. It was quicker and the usage learning curve was not as steep.

3.9.7 User Feedback

The apparent interaction control from the user point of view is nearly null. Users requested some sound markers or graphical clues to inform them about the system state.

Task easiness Perceived ease in performing a task is considered an important factor influencing user experiences when interacting with a given technology or system [197]. The ease or difficulty of the task was measured by a Single Ease Questionnaire (SEQ) right after participants finished it. The SEQs rely on a seven-point scale that ranges from 1 (very difficult) to 7 (very easy).

As shown by the boxplots in Figure 3.15, most of the VUI Tasks were perceived to be rather easy by participants, independently of the considered setup

(all median SEQ scores were equal or above 4). "Entering sleep data" was described as more difficult than other VUI tasks; more issues in vocal interaction occurred during this task, independently from the task itself. Regarding the touch-only tasks, participants perceived them as more difficult than the speech-based tasks (all median SEQ scores were equal or under 4).

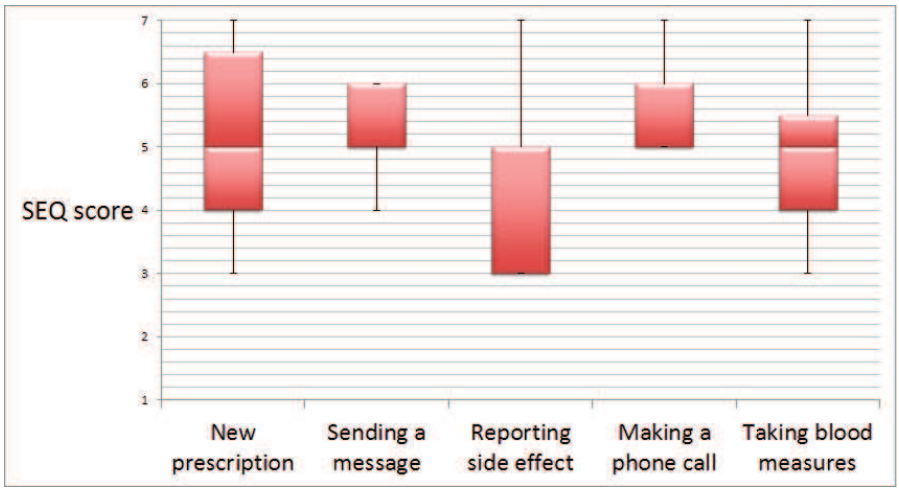


Figure 3.15: Result of the SEQ

Usability The System Usability Scale (SUS) was employed to measure the usability of the entire vAssist prototype system. SUS scores fall between 0 and 100, the higher the score the better the overall usability. Scores below 50 are typically considered not acceptable [172]. The SUS scores for Austria and France were 68 (sd = 17.2) and 70 (sd = 11.5), respectively. Following Sauro and Lewis [172], who suggest classifying SUS scores according to American school grades (from A [=excellent] to F [=failure]), the Austrian SUS-score equals grade "D" and the French SUS-score grade "C". This indicates the need for improvement of the Apps used within vAssist in terms of usability for both countries.

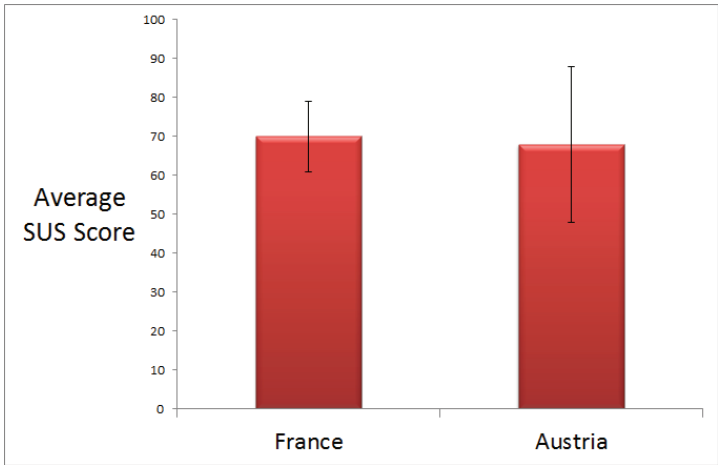


Figure 3.16: Result of the SUS

Speech assessment The Subjective Assessment of Speech System Interfaces (SASSI) questionnaire was employed to measure the interaction quality of the speech-based vAssist services. The SASSI is a tool to evaluate speech-based interfaces [83]. Users are presented a series of statement, which they grade according to their agreement with them, on a scale from 1 to 7. The analysis of the questionnaires provides developer with an assessment of the system along several axes such as the easiness, the friendliness, the speed, etc.

The SASSI scores for Austria are summarized in Figure 3.17, those for France in Figure 3.18. The higher the values on a subscale, i.e. "Response Accuracy, "Likeability", "Cognitive Demand", and "Speed", the more participants agreed that the system performance was good as opposed to bad (vice versa for lower values). Note that the SASSI scores reported based on only those tasks where speech-based interaction was enabled. For the Austrian sample, the results indicated that both "Performance Accuracy" and "Speed" were judged to be neither good nor bad, i.e., neutral in agreement, and that "Likability" and "Cognitive Demand" were judged to be fair, i.e., slight agreement.

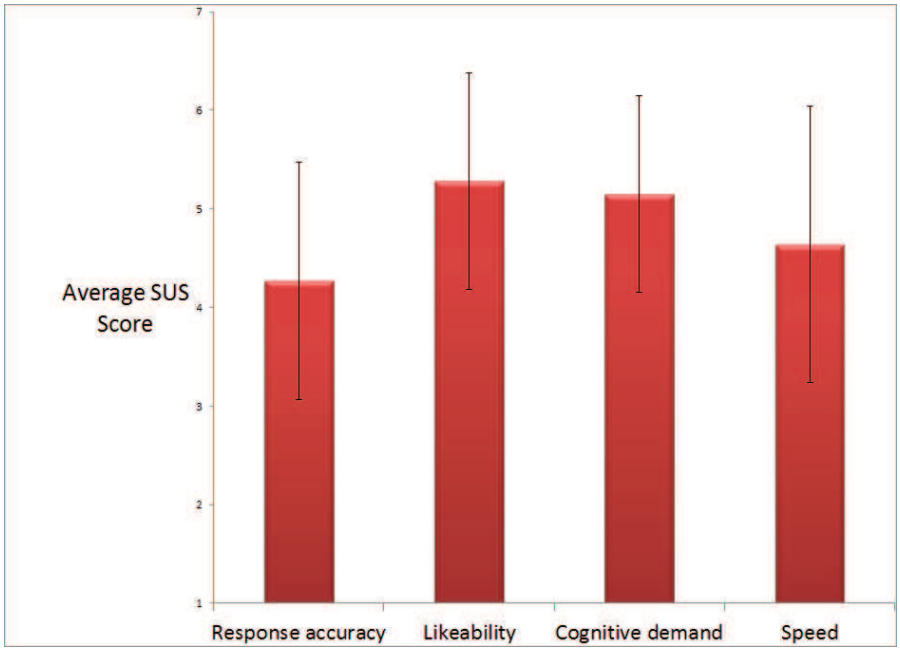


Figure 3.17: Result of the SASSI for Austria

For the French sample, the results indicated that "Likeability" and "Cognitive Demand" were judged to be fair, i.e. slight agreement. In other words: participants appeared to like the system and were not overwhelmed by its cognitive demands. By comparison, "Response accuracy" and "Speed" were judged to be neither good nor bad, i.e. neutral in agreement.

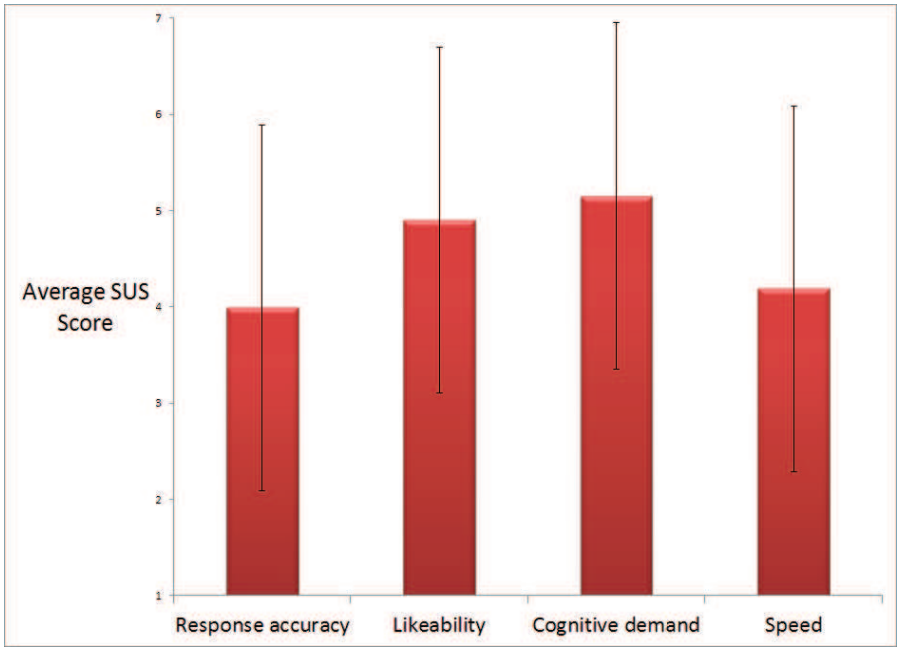


Figure 3.18: Result of the SASSI for France

Overall the likability and the cognitive demand were judged as good according to the SASSI. The average evaluation results were, respectively, 5.28/7 and 5.15/7.

3.10 Conclusion and Future Work

In this chapter, two methods for continuous listening applied to SDSs were presented.

The first method was isolated within the system. Thus, while the SSA had the knowledge of the application domain, there was no real-time feedback to augment the recognizer KSs nor any context inclusion in the semantic representation of the user's intent.

Also the second method, i.e. the method tested with a hand-held device, suffered from the lack of control on the client side. The ability to set the volume level and to mute/unmute the microphone (pause the call) were not available neither was the multi-modality feature.

Overall, the automatic speech recognizers are analyzing the "raw" signal. Some signal processing algorithms could have been applied in order to "clean" the signal from the ambient noise or to normalize it, thus reducing the effect of the speaker's distance on the signal amplitude.

In the mobile robot setup, the way the sound was captured was far from

optimal. Some experiments led by other teams working in the domain showed that the recognition rate benefits from the use of several microphones located in strategic spots associated with an appropriate selection algorithm [102, 103].

Additionally, the CMT developers worked on a speaker tracking feature based on the three directional microphones available on the device.

The sounds/speech analysis system was pushing messages on a server which every devices or services in the house were connected to. Making use of this real-time information flow would definitely have helped in the implementation of a more reliable sound processing system.

The second setup, which specifically targets hand-held devices, is currently the method used in the vAssist project, so are the methods exposed next, for the interpretation of the user utterances and the modeling of dialogs.

Chapter 4

A Sub-system to Map Natural-language Utterances to Situated Parameterized Dialog Acts

4.1 Introduction

The NLU component of an SDS acts as the link between the wide informal real-world language space a user speaks in and the well-structured messages that a computing machine can process [136, 196]. Whether its inputs are transcription hypotheses from an ASR engine or typed on a keyboard, the automatic system needs to convert the inputs into structures bearing the meaning or the intent of the user in order to "understand" her/him. An NLU system produces DAs for the DM to work with [168].

Several methods have been used to convert either a word sequence, a word lattice or a (ranked) set of sentences to such meaningful data structures [9, 10]. There is however no universal agreement regarding the kind of data an NLU system should produce and the data type it should take in [2, 160].

While the process is uni-directional, from NL to meaning representation, its algorithm integrates and combines the information from several sources into the algorithm.

This chapter provides an overview of the issues and challenges the NLU domain currently faces. The platform introduced earlier integrates a set of inter-connected components to transform NL spoken utterances to DAs, the unit of dialog, so that the DM may understand the user and react accordingly. Since the NLU is not a single piece of software but a set of independent services, each one of them will be described in a specific section.

4.2 State-of-the-art Methods for NLU

Over the years, many techniques to interpret the content of the speech signal have been proposed [45, 48]. This section presents a selection of the state of the art in these techniques.

4.2.1 Keywords Spotting

Early NLU systems based their analysis on keywords [48, 204]. The inputs' textual content was searched for words which, by themselves, defined the intent or the signification of the user's turn.

The keywords spotting method limits the size of the application domain. Indeed, the overlap between utterances has to be minimal and the keywords have to be manually defined.

4.2.2 Context-free Grammars

Definition

A Context-Free Grammar (CFG) [53, 132, 201, 202, 203] is a tuple $G = (S, N, \Sigma, R)$, where:

- S is the start symbol
- N is a set of non-terminal symbols
- Σ is a set of terminal symbols, also called the alphabet
- R is a set of derivation rules

Given a terminal symbols sequence, drawn from Σ, the objective is to build a parse tree, whose root is S, applying the derivation rules.

Parsing algorithms

The derivation algorithms are characterized along two axes: the parsing direction and the priority.

The next paragraphs will be based on the following CFG instance:

- $S = P$
- $N = \{P, V, C, D, N, PP\}$ (P = Phrase, V = Verb, C = Complement, D = Determiner, N = Noun, PP = Preposition)
- $\Sigma = \{$the, cat, dog, bird, sleeps, runs, eats, on$\}$
- $R = \{$

  $P \rightarrow SV \mid SVC$

  $S \rightarrow DN$

$C \rightarrow S \mid PPS$

$D \rightarrow$ the

$N \rightarrow$ cat | dog | bird

$V \rightarrow$ sleeps | runs | eats

$PP \rightarrow$ on

}

Let us use the above algorithm and try to obtain a parsing for the terminal symbols' sequence "the cat sleeps", applying the grammar.

Top-down leftmost parsing

Starting from the start symbol, the algorithm applies the first derivation rule available. For instance, the first rule is $P \rightarrow SV$.

The next table details the top-down leftmost parsing steps for the terminal symbols' sequence: "the cat sleeps". The algorithm attempts to expand the tree applying the leftmost derivation rule in R until it reaches terminal symbols from Σ. If a suitable tree can not be found, a backtracking mechanism tries out alternative rules.

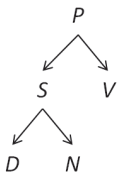
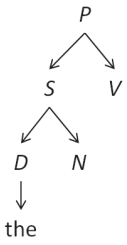
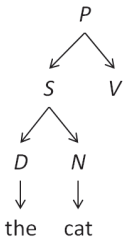
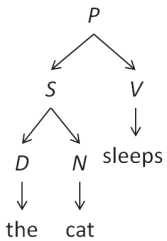
Applied derivation rule	Stack state	Parse tree
Start symbol: P		
$P \rightarrow SV$		
$S \rightarrow DN$		
$D \rightarrow$ the		
$N \rightarrow$ cat		
$V \rightarrow$ sleeps		

Table 4.1: Example of top-down leftmost parsing of the sequence: "the cat sleeps"

Top-down rightmost parsing

In the case of a rightmost algorithm, the resulting parse tree is identical. The applied sequence of derivation rules is:

$P \to SV$

$V \to$ sleeps

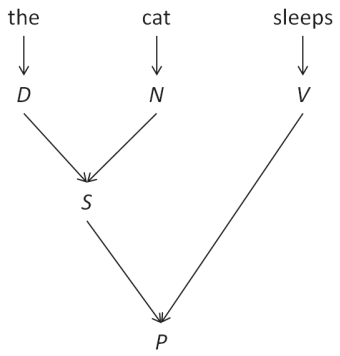
$S \to DN$

$N \to$ cat

$D \to$ the

Bottom-up leftmost parsing

A bottom-up parsing algorithm applies the derivation rules from the symbols' sequence to the start symbol. From the same example sequence, one can build the following tree:



Top-down rightmost parsing

One obtains the same parse tree using a rightmost priority. The sequence of rules to apply is:

sleeps $\to V$

cat $\to N$

the $\to D$

$DN \to S$

$SV \to P$

Advantages and drawbacks

CFGs are appropriate to quickly specify a small set of utterances from a formally-defined language. However, the hand-crafting requirements as well as the inflexibility of the parsing restrict their usage.

4.2.3 Probabilistic Context-free Grammars (cf. 3.2.4)

Probabilistic Context-Free Grammars (PCFGs) [68] extend from the CFGs with the addition of the uncertainty in the parsing and a method to resolve ambiguities that may occur in the selection of derivation rules.

The definition of a PCFG is similar to the one of a CFG. It's a tuple $G = (S, N, \Sigma, R)$ where:

- S is the start symbol
- N is a set of non-terminal symbols
- Σ is a set of terminal symbols, also called the alphabet
- R is a set of derivation rules

Additionally, each derivation rule has an associated probability and, for the same left symbol, all the probabilities sum up to 1.

An example of such probabilistic rules, adapted from the previous example would be:

$R = \{$

$P \rightarrow SV(0.5) \mid SVC(0.5)$

$S \rightarrow DN(1)$

$C \rightarrow S(0.3) \mid PPS(0.7)$

$D \rightarrow \text{the}(1)$

$N \rightarrow \text{cat}(0.5) \mid \text{dog}(0.25) \mid \text{bird}(0.25)$

$V \rightarrow \text{sleeps}(0.5) \mid \text{runs}(0.2) \mid \text{eats}(0.3)$

$PP \rightarrow \text{on}(1)$

$\}$

Parsing

The derivation algorithms are the same as the CFG's. Additionally, when more than one parse tree can be built for a sequence of terminal symbols, the likelihood of each one of them is computed with:

$$P(tree) = P(R_1) \times P(R_2) \times ... \times P(R_n)$$

Where $P(tree)$ stands for the overall probability of a parse tree, $P(R_x)$ is the probability associated with the derivation rule x and $tree$ results from the application of the rules R_1 to R_n.

The tree with the highest overall probability is selected.

Learning

The probability of the rules is learned from an annotated corpus. The following formula applies:

$$P(X \rightarrow Y) = \frac{c(X \rightarrow Y)}{c(X \rightarrow ?)}$$

Where $P(X \rightarrow Y)$ is the probability of the rule that derives Y from X, $c(X \rightarrow Y)$ is the count of occurrences of that rule in the annotated corpus and $c(X \rightarrow ?)$ is incremented every time the symbol X is expanded in the training source annotation trees.

Advantages and drawbacks

The extension of the CFGs with stochastic rules does not alleviate the need for handcrafting the grammar components. While the probabilities are learned automatically, making the method more powerful, the burden of creating the initial grammar and annotating a set of examples hinders the process.

4.2.4 Hidden Markov Model

Chronus [144, 145], a parsing system developed at the AT&T Bell laboratories, proposes to apply HMMs to the semantic labeling task [12, 178]. The HMM's hidden states define the semantic labels and generate word observations.

A description of the HMMs is included in section 3.2.6. Figure 4.1 illustrates the method applied to semantic parsing.

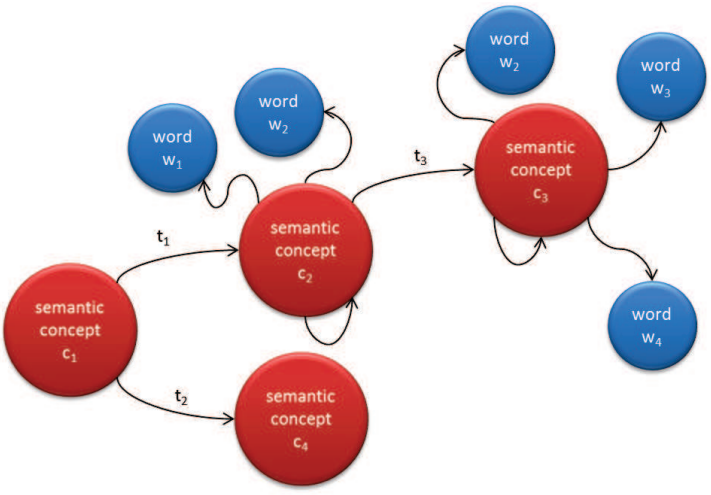


Figure 4.1: Illustration of semantic parsing using HMMs

Learning

A corpus of labeled natural-language word sequences is provided to train the model. The annotations are a semantic label, i.e. an HMM state, for each word.

The forward-backward algorithm defines the HMM's parameters so that it maximizes the likelihood of the dataset.

Decoding

As for any other HMM-based decoder, the Viterbi algorithm efficiently searches for the most likely sequence of hidden states in the HMM that produces the observation sequence, i.e. the sequence whose overall probability is the highest.

Advantages and drawbacks

Such a paradigm enables a model to be learned from annotated data. Hence, actual manual work is reduced to the collection and the annotation of a corpus.

However HMM-based parsers build flat trees which do not catch distant dependencies. Moreover, especially in the case of spontaneous speech, a full parsing, where every word is associated to a label, does not generalize well. Adding filler words to the training corpus improves the scope of the parser. Still, we argue here that the partial parsing of utterances is more appropriate for NL SDSs.

4.2.5 Hidden Vector State Model

The HVS model extends the HMM-based parser [76, 77, 78, 180]. Since the former lacks a hierarchical labeling ability, the HVS model aims at improving

it such as to allow a multi-level parse tree to be generated.

A parse tree can be represented as a sequence of vectors built from the chains of labels, from the preterminal one to the root one. An example of such a sequence is shown in Figure 4.2 with the corresponding tree.

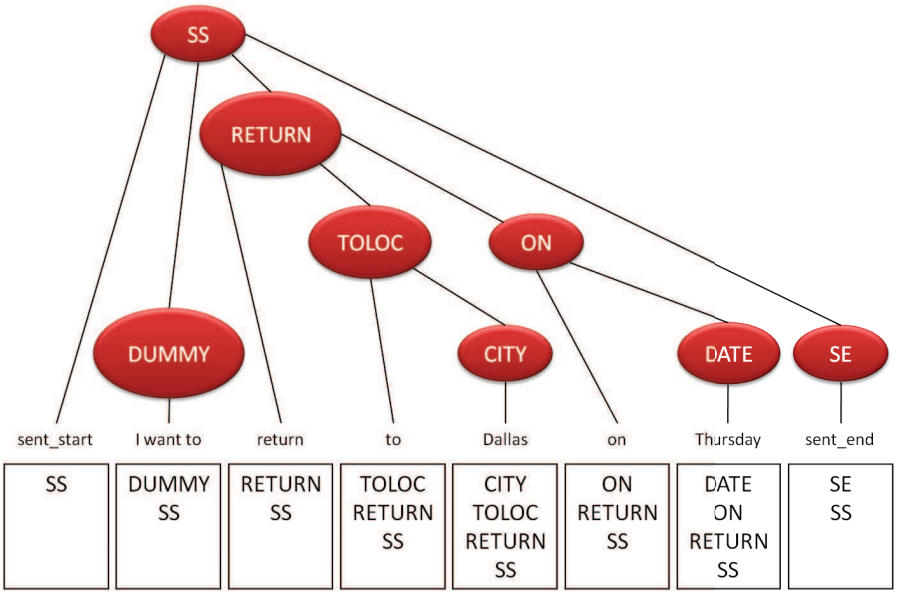


Figure 4.2: Parse tree representation as a sequence of vector states (source: [76])

SS, DUMMY, RETURN, TOLOC, ON, etc. are semantic concepts. For instance, SS is the sentence start marker, SE is the sentence end marker, RETURN is the return trip information, TOLOC is a location concept, etc.

The vectors encode a sequence of stack states. Therefore, the transitions essentially use two operations: popping concepts out of the stack and pushing some in.

Consequently, a corpus can be annotated to train an HVS model, i.e. an HMM whose hidden states are semantic label vectors generating words. Such a model is then defined by:

- $W = w_1, w_2, ..., w_m$ a set of words
- $C = c_1, c_2, ..., c_n$ a set of vectors of concepts $c_i = [c_i[1], c_i[2], ..., c_i[D_i]]$ where D_i is the size of the vector c_i. These are the model's states.
- $A = a_{11}, a_{12}, ..., a_{nn}$ is a set of state-transition probabilities $a_{ij} = P(c_j|c_j)$
- $B = b_1, b_2, ..., b_n$ a set of output probability distributions $b_i(w_x) = P(w_x|c_i)$

The probability of a sequence of concept vectors C, given a sequence of words W is given by:

$$P(C|W) = P(w_1|c_1) \times P(c_1) \times \prod_{t=2}^{T} P(w_t|w_1, w_2, ..., w_{t-1}, c_t) \times P(c_t|c_1, c_2, ..., c_{t-1})$$

where $P(w_t|w_1, w_2, ..., w_{n-1}, c_t)$ is the probability of observing the word w_t while in state c_t after observing w_1 to w_{t-1}. $P(c_t|c_1, c_2, ..., c_{t-1})$ is the probability that the state c_t is entered after c_1 to c_{t-1} have been visited.

Tractability

Computing the parameters of the HVS model is often intractable without further constraints on the state transitions and the vectors' size.

To reduce the search space, the transitions are split into two constrained operations:

- The stack is shifted by 0 or more symbols
- Only one new symbol is pushed into the stack

The word observations are assumed to be dependent only on the emitting state.

Thus, the formula becomes:

$$P(C|W) = P(w_1|c_1) \times P(c_1[1]) \times \prod_{t=2}^{T} P(w_t|c_t) \times P(c_t[1]|c_t[2...D_t]) \times P(n_t|c_{t-1})$$

where $P(w_t|w_1, w_2, ..., w_{n-1}, c_t)$ becomes $P(w_t|c_t)$ and $P(c_t|c_1, c_2, ..., c_{t-1})$ becomes $P(c_t[1]|c_t[2...D_t]) \times P(n_t|c_{t-1})$. $P(n_t|c_{t-1}$ is the probability of the position shift of the stack and $P(c_t[1]|c_t[2...D_t])$ is the probability of the single concept to be pushed onto the stack.

Advantages and drawbacks

The HVS model is able to capture hierarchical structures in the parsing while keeping the computation cost low enough to stay tractable. However, this model applies a full parsing to the data which then, as shown before, may not be ideal for spontaneous speech, which favors the partial parsing of the NLU task.

4.2.6 Semantic Frame: A Meaning Representation

An SF is a data structure to represent the semantic content of a user utterance or interaction move [21, 50, 123, 124]. It has a 2-level hierarchy.

The goal, i.e. the implicit or explicit intent of the entity emitting a signal, is characterized by 0 or more slots. Slots are pairs, a name and a value, defined in a multi-dimensional semantic space.

Figure 4.3: A SF example

Figure 4.3 shows an SF. This is the semantic representation of the input of a user asking for the weather forecast in Paris on the 15th of November, 2012. For instance, the spoken utterance may be "What is the weather like in Paris now?", "Will it be sunny in the French capital on the 15th?" or "Show me the weather". Within the system's scope, all those inputs carry the same meaning. They may result, after being processed by an NLU system, in the SF shown above.

4.3 Natural Language Understanding: Issues and Challenges

The review of the state of the art in the NLU domain highlighted both the limit of the current method and the requirements for those systems [13, 14, 51, 52, 56, 159, 187]. This section presents an analysis of both aspects. On the one hand, what is mandatory for an NLU sub-system to implement and, on the other hand, what are the most prominent issues when building such components.

4.3.1 Challenges

Keep a mixed initiative

Dialog designs can be characterized along an initiative scale that ranges from user-driven to machine-driven initiative.

In the case of a machine-driven dialog users strictly follow the predefined flow of the system. Interaction turns that are outside the scope of the dialog are not understood. They may either be discarded or, in the worst case, might lead to a system failure. Despite this, a machine-driven design makes the dialog easier to control and therefore less prone to errors. Yet, due to the lack of adaptability exposed by the system, it appears less human-like.

On the other end of the scale, full user-driven dialog designs minimize the functional range of a system as they only require the DM to react to commands. Its role of maintaining a dialog state to select the best next step, is not relevant anymore. It is the so-called command-and-control paradigm.

Advanced SDSs therefore aim at implementing a mixed initiative, where the system's integrity and its goals are sufficiently defined. The user, however, should not be restricted by the type and amount of spoken utterance he/she can use to interact.

In order to maintain a mixed initiative, the NLU component(s) of advanced SDSs is(are) required to handle a wider vocabulary, various speaking styles, spontaneous speech and need to implement some recovery methods so that the system may react appropriately to everything the user may attempt.

Moreover a gradual fallback mechanism is needed for the system, so as to signal out-of-scope cases, attempt inner repairs and/or guide the user back into the scope of the system.

In other words, a (true) mixed initiative means that the user is not restricted to a set of utterances he/she can speak in any given state. Rather, the system is able to select and discard respectively the relevant utterances and the out-of-scope ones.

Allow for variability

In the past, SDSs were primarily command-controlled. They applied small and static grammars and used keyword spotting to identify the user's intention. Hence, users usually had to be trained before interacting with the system so that they would stay within the scope of its understanding capabilities while interacting.

More recent NLU systems, however, target the casual speaker using a NL to express its intent.

"A natural language or ordinary language is any language which arises, unpremeditated, in the brains of human beings. Typically, therefore, these are the languages human beings use to communicate with each other, whether by speech, signing, touch or writing. They are distinguished from constructed and formal languages such as those used to program computers or to study logic" [118].

Understanding such an unrestricted communication language is achieved through relaxing the constraints put on the grammar and the words of utterances as well as further processing of the parsing results.

Humans naturally make use of co-references, ellipses, shared knowledge, etc, in a spoken interaction. Thus, NLU systems, in the quest to build them closer to the human abilities, should allow for as many ways of expressing an equivalent intent as a human may utter it. Such would allow for more variability with respect to possible user inputs and therefore may lead to a more human-like system.

Disambiguate using the dialog history

To optimize the exchange of information, a human-human dialog creates a shared knowledge space between dialog participants. As for an SDS, this inherent grounding process is similar to keeping track of the dialog history.

In order to disambiguate references to earlier spoken entities or conversation topics, the NLU components should have access to these recordings. This would allow for mimicking the human-human interactive build-up [49].

Disambiguate using the dialog expectations

The SDS's DM expects formal meaning representations to be converted to actual dialog moves or DAs similar to parametrized dialog commands. A DA is the unit of action in the dialog flow.

The set of DAs that are available is finite, dynamic and depends on the current state of the dialog[1]. At every turn, a list of these can be extracted from the DM. It contains the actions the manager is able to handle, possibly with a probability or ranking associated with them. This defines the expectations. One can think of them as constraints to the inputs or as guidances.

An NLU system aims at mapping the NL inputs to one or several action unit(s) made available by the DM. The tying between the initial infinite (user's) set and the finite well-defined DM's DAs list is either complete, i.e. there is always a unique mapping from input to DA or partial with a default "non understanding" action which is produced when no coherent mapping can be extracted. Also, the mapping method can be based on static rules or on dynamic links synchronized to the state of the dialog, a particular user, etc.

Two input utterances carrying the same meaning may lead to different decisions depending on the dialog state. The right action, i.e. the real DA, is to be determined by the NLU component.

Use the environment knowledge

SDSs should not be closed boxes whose resources are restricted to static databases, dialog models and/or policies. Today's ubiquitous systems are often "aware" of their surrounding environment, and possibly share this awareness with the user. This ongoing trend towards multi-modal awareness, where external sensors increasingly deliver real-time information to a range of different systems, may also be used to improve language understanding. Additional data, if successfully processed, can lead to a better context disambiguation and consequently increase the overall NLU accuracy.

Here is an example of using the inner clock of a system to inject some environment knowledge into the dialog. It can be found in many online automatic services such as train and plane booking, navigation systems, etc. Let us consider the concept "now", referring to the current point in time.

A user of, for instance, a travel booking SDS system declares that he/she wants to travel "now". With a sufficiently good parser, one gets an SF whose goal is *traveling* and whose only parameter is *travel_time = now*.

The concept "now", before being passed on to the DM, has to be resolved to represent an absolute point in time. For that, an NLU component may call the host's clock functionalities, associated with a proper formatting method to transfer "now" to *travel_time = July 22, 2014. 3.44pm*.

Many other environment sensor's readings affect the meaning of an utterance. NLU systems should use these information sources as much as possible.

[1] Here a state does not refer to a 'real' state, such as the ones used in MDPs or POMDPs, but rather to the status of the dialog

4.3.2 Issues

Several issues have been identified which increase the difficulty level of the NLU task from straightforward to extremely difficult.

Filler words and backchannels

Human conversations contain filler words such as “euh”, “hum”, “so” and backchannel interjections such as “ah”, “OK” which serve a purpose in the information exchange in a human-human conversation [60]. In the case of a multi-modal interaction, the backchannels can also be gestures, head nods, gazes or anything that conveys information about the activity of the listener.

In order to closely mimic human behavior, handling and processing them is necessary.

Elliptical construction

An ellipsis occurs when one or more words of an utterance are omitted, yet it is still understandable given the context of the interaction [26]. Missing tokens can refer to past dialog topics or to common background knowledge.

Processing such constructions requires first to detect elliptical constructions, retrieve the missing information and integrate it back into the turn. Each one of these three steps is a tricky challenge to tackle.

Indirection

The best way to explain what is an indirection is to consider a dialog example. Here are two:

A: The phone is ringing.	*Could you answer the phone?*
B: I’m in the bath.	*I can’t pick up the phone.*
A: OK.	*I’m going to answer the phone.*

A: I’m hungry.	*I want to eat something. Please cook for me.*
B: What would you like to eat?	*What would you like to eat?/What can I cook for you?*

The italicized sentences are the actual social interpretation of the spoken utterances on the left-hand side. Here, the social interpretation differs from the word-content interpretation of the inputs [65]. They require several degrees of abstraction from the word content.

Anaphoras

Anaphoras are references to expressions or clauses in the context but use a different wording to refer to them [65]. Here are two examples:

I'm going to my friend's place to visit ***her****.*
her refers to the expression *my friend*.

Since ***he*** *is back, John wants to be a cook.*
The pronoun ***he*** refers to *John* as resolved in the second clause.

4.4 Platform's NLU System Overview

The current version of the SDS platform integrates an NLU system, as shown in the black shape in Figure 4.4

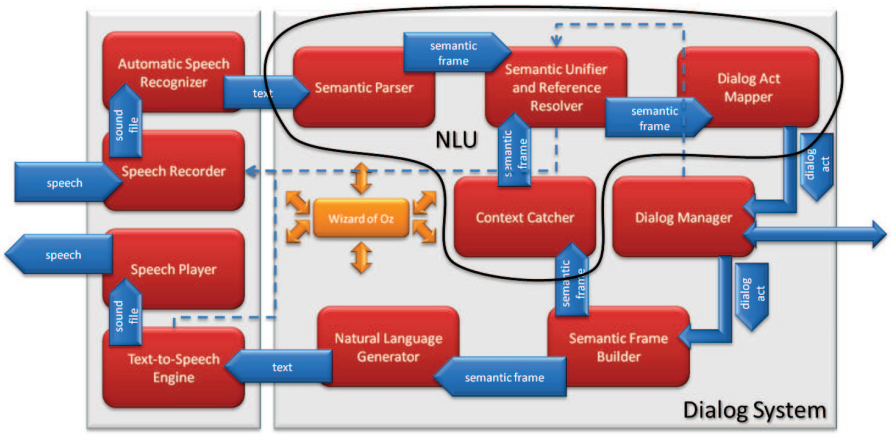


Figure 4.4: The platform's NLU sub-system

This NLU sub-system is a set of inter-connected components, such as in [11, 45] providing a mapping from transcribed utterances to parameterized DAs related to the current dialog state. As such it is a multi-step dynamic process with clearly separated roles taken on by individual sub-components.

The input space size of any sub-component is greater or equal to the size of its output space. In other words, the NLU engine processes user utterances sequentially and, at every stage, the abstraction level of the semantics extracted from the text becomes more machine oriented. This sequential analysis of the inputs is illustrated in Figure 4.5.

Figure 4.5: The one-way interpretation process of spoken user utterances

4.5 Semantic Parsing

The SP, which gets inputs from the ASR, associates semantic labels to text utterances (or parts of them). The most commonly used parsing techniques are based on CFGs or PCFGs, which are either hand-coded, based on the analysis of collected dialog data, or designed by experts.

The SP, here, integrates the algorithm proposed by [92], which is the application of the work from [23, 95]. Instead of matching whole sentences with parse structures, the algorithm looks for patterns in chunks of the text-level utterance and in the temporary (i.e. currently assigned) SF.

The module applies an ordered set of conditional rules, which is learned from data.

4.5.1 Training

The training of the SP is based on a corpus consisting of annotated utterances. The annotations are, for each utterance, the SF containing all the concepts extractable from the corresponding word utterance. The main concept, i.e. the overall intent conveyed by the utterance, is the goal of the SF.

The algorithm, given all those pairs (utterance, SF), searches for the best sequence of rules to convert default SFs to the ones in the corpus.

Transformation rules A rule has two parts. The trigger part is a pattern to search for in the utterance as well as in the temporarily assigned SF. Triggers are combinations of:

- an n-gram in the word utterance
- a skipping 2-gram in the word utterance
- the goal of the temporary SF
- the name of a slot
- the name and value of a slot

The operation part transforms the temporary SF. It is a combination of:

- a goal substitution
- a slot addition
- a slot removal
- a slot's value substitution

Algorithm The default SF is the frame whose goal is occurring the most in the training corpus. The training starts with every utterance assigned to it. At every step, the Levenshtein distance is computed to measure how close the result of the application of a rule sequence is from the "truth", given by the annotated corpus.

Every available rule, given the state of the database, i.e. the temporary transformed initial corpus, is evaluated and the distance to the reference corpus is computed. The one that reduces the distance the most is applied to the database, creating a new state for it, and is added to the rule list. As long as the algorithm can find a rule that "improves" the database above a given threshold, the algorithm executes. When no additional rule can be applied to narrow the distance, the result is an ordered set of rules.

4.5.2 Decoding

This phase is less complex than the training part. An utterance to parse is first labeled with the default SF. This is the SF with no slots and whose goal is the one occurring the most in the training corpus. It is the first operation of the rules sequence whose condition is always true. Rules are taken from the list produced by the training. The trigger of each of them is matched against the current utterance and the temporary SF. The rule is applied, i.e. the transformation operates, if and only if every condition in the trigger is fulfilled. Iteratively the initial SF is transformed to the expected one, given the input to the parser.

Below is the log of parsing the utterance "I would like to call John" to illustrate this paragraph.

```
Input content: I would like to call John
Decoding rules file: /opt/TPT/conf/vassist/sp/sp_corpus_vassist_en.pckl-decoder
Trigger conditions: {'nearestDepTreePOSWord': 0, 'nSlots': 1, 'nStarGrams': 4,
    'DBItems': 'replace', 'hasSlots': 1, 'nGrams': 4, 'useDeps': 0, 'speechAct': 1}
173 rules
12551 slot values
Dialog act: I would like to call sv_name-0 . x 0 slots
Generated grams count: 25
Decoder loaded, applying rules...
Rule[Trigger(N-Gram: None - Speech Act: None - Slots: None - HasSlots: None),
    Trans(SpeechAct: input - AddSlot: None - DelSlot: None - SubSlot: None)]
Rule[Trigger(N-Gram: ('sv_name-0',) - Speech Act: None - Slots: None - HasSlots: None),
    Trans(SpeechAct: None - AddSlot: name=sv_name-0 - DelSlot: None - SubSlot: None)]
Rule[Trigger(N-Gram: ('call', 'sv_name-0') - Speech Act: None - Slots: None - HasSlots: None),
    Trans(SpeechAct: make_a_call_dialog - AddSlot: None - DelSlot: None - SubSlot: None)]
Semantic frame: goal=make_a_call_dialog;name="John"
```

Table 4.2: Log of the decoding of "I would like to call John"

4.5.3 Slot Values Clustering

There is an additional feature to reduce the complexity of the initial corpus. For example, let us assume that one would like to set up the parser for utterances requesting the system to call someone on the phone. Three sentences are collected and annotated. That is:

let's call John	[goal(call);slot(callee=John)]
please call Jack	[goal(call);slot(callee=Jack)]
ring James	[goal(call);slot(callee=James)]

Table 4.3: SP training corpus 1

A parser trained with this tiny corpus has four rules:

Rule#	Trigger	Operation
1	-	goal=call
2	utterance contains 1-gram "'s"	slot(callee=John)
3	utterance contains 2-gram "please call"	slot(callee=Jack)
4	utterance contains 1-gram "James"	slot(callee=James)

Table 4.4: Ordered set of rules 1

Obviously, this is not valid. Applying these rules, a user saying "Let's call Jack" would end up speaking to John while another user asking "Please call John" would reach Jack. To solve this, the corpus can be multiplied so that the three utterances are tuned to the three callers, i.e.:

let's call John	[goal(call);slot(callee=John)]
please call John	[goal(call);slot(callee=John)]
ring John	[goal(call);slot(callee=John)]
let's call Jack	[goal(call);slot(callee=Jack)]
please call Jack	[goal(call);slot(callee=Jack)]
ring Jack	[goal(call);slot(callee=Jack)]
let's call James	[goal(call);slot(callee=James)]
please call James	[goal(call);slot(callee=James)]
ring James	[goal(call);slot(callee=James)]

Table 4.5: SP training corpus 2

Then the parsing produces four rules:

Rule#	Trigger	Operation
1	-	goal=call
2	utterance contains 1-gram "James"	slot(callee=James)
3	utterance contains 1-gram "Jack"	slot(callee=Jack)
4	utterance contains 1-gram "John"	slot(callee=John)

Table 4.6: Ordered set of rules 2

To avoid this increase in corpus size and the burden of collecting, annotating and checking more data, the algorithm can be augmented with a slot database.

The base organizes some tokens according to two levels of labelisation. Figure 4.6 is a schematic of the database structure.

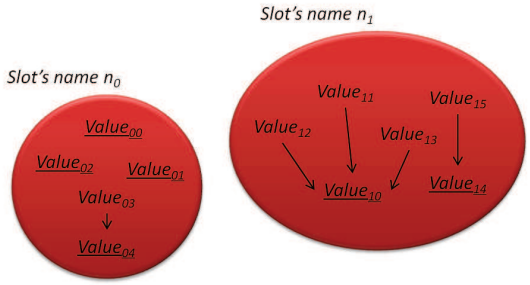


Figure 4.6: Structure of the slot database

A token, if contained in the database, is labeled with the name of the slot n it belongs to. Furthermore, the token is either the member of a slot's value subgraph or the head token (arrows are pointing towards head values). These tiny subgraphs are synonyms.

A database for our example above is schematically presented in Figure 4.7.

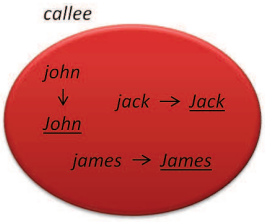


Figure 4.7: Example slot database

The learned list of rules is reduced to only two rules:

Rule#	Trigger	Operation
1	-	goal=call
2	utterance contains 1-gram "callee-x"	slot(callee=callee-x)

Table 4.7: Ordered set of rules 3

Adding a slot's value to the database extends the scope of the parser easily.

4.6 Semantic Unifier and Reference Resolver

In this section, we present a novel component: the Semantic Unifier and Reference Resolver (SURR). This versatile component holds a pretty simplistic forest

of nodes and is used to mine the dialog history, incorporate external sources of information and add some local turn context. It is the meeting point of three different sources, as schematized in Figure 4.8.

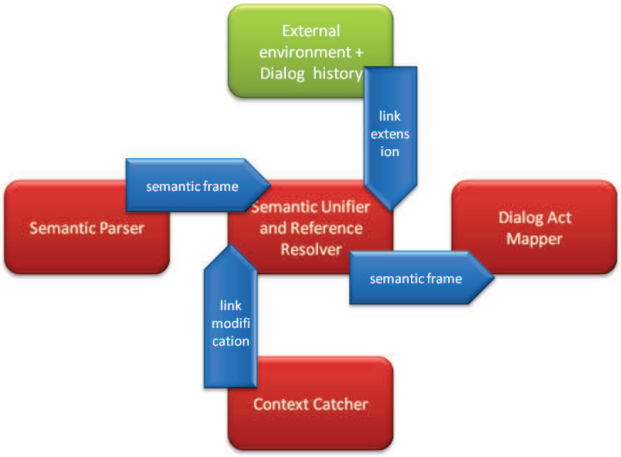


Figure 4.8: SURR connections

The internal structure of the SURR is shown in Figure 4.9

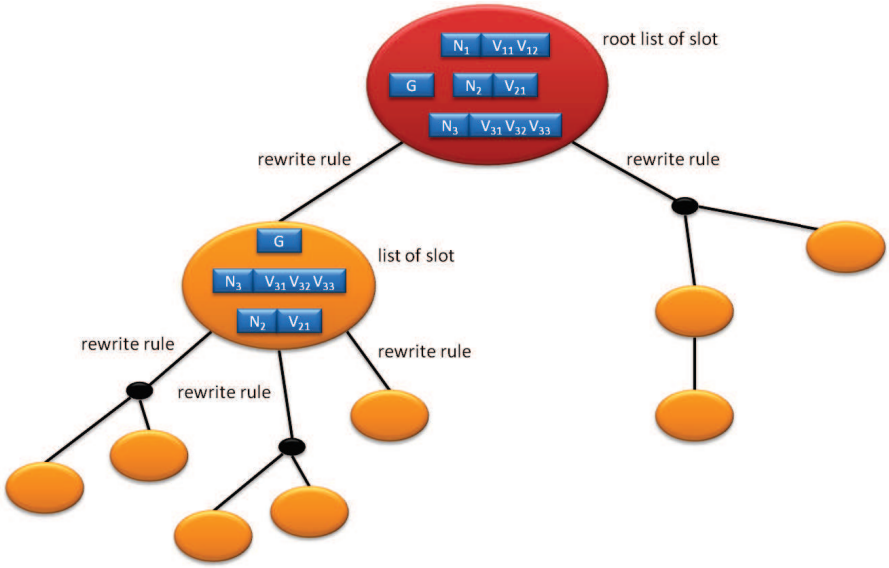


Figure 4.9: SURR knowledge representation

The SURR embeds a forest structure at its core. Trees consist of hierarchies

of fully or partially defined SFs, some nodes are also calls to external systems, APIs or services. When requested, the SURR is able to dynamically modify (remove/add) branches to any part of the forest. The top node of a hierarchy defines a root node (in red in the drawings).

The SURR algorithm tries to find a unique path from an input SF, i.e. from the parsed user input, mapped to nodes of the forest, to a root node. Going up the trees, the algorithm applies the optional operations held on branches. These operations can have a single SF input argument (Figure 4.10)

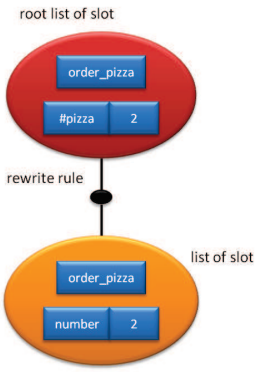


Figure 4.10: Single-argument operation

Or may request several ones (Figure 4.11)

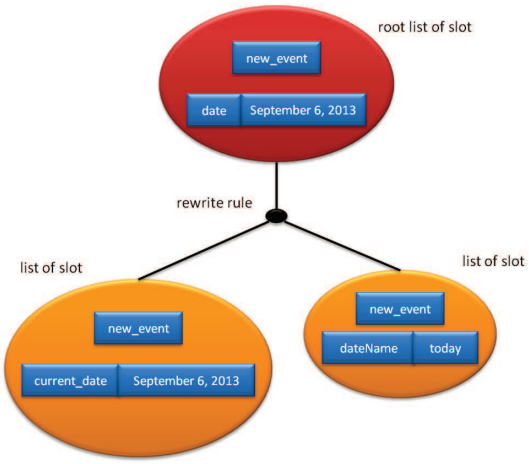


Figure 4.11: Multiple-arguments operation

A root node reached is equivalent to the input of the user being contextualized. In case the algorithm can not find such a path, i.e. the SURR has

failed to produce a suitable SF (given the current context and knowledge available), a "NoMap" SF is generated to signal to the next components that a non-understanding occurred.

4.7 Context Catcher

The complement of the external knowledge extractor (see Section 4.8.3) is the Context Catcher (CC). This component analyzes the output of the DM and subsequently updates the SURR, i.e. dynamically changes its inner structure, to match the dialog context.

The component relies on a set of classes which hold some slots requested by the DM. When a request to such a slot name is caught by the CC, it creates a link from the generic class name to the more specific slot name, such that an input labeled as generic by the parser could be resolved to the requested slot. The CC also erases the temporary associations once the expected concept is not in focus anymore.

Here again, an example to illustrate that: the dialog application is a pill box management tool, i.e. an application that helps to manage the treatement of a person, gives reminders and monitors the intakes (cf. 2.7.2). One of the services is to add an aspirin[2] prescription from a practitioner to the box. The scenario:

User U1: *Hi, I just got a new prescription that I'd like to add.*
System S1: *What is the name of the drug?*
User U2: *That is for aspirin.*
System S2: *How many intake a day?*
User U3: *2 intakes.*
System S3: *How long between intakes*
User U4: *6 hours.*
System S4: *How many pill for each intake?*
User U5: *Just 1*
System S5: *What time is the first intake?*
User U6: *At 9 am.*
System S6: *Please check the information on the screen. Are they correct?*
User U7: *Hum... Yes.*
System S7: *Good. The new prescription is saved. You will get reminders for it.*
User U8: *Thanks!*
System S8: *You're welcome.*

Let's focus on the turn 4 and 5. the information carried by the utterances are weak. For the first one, the ideal SP would produce Figure 4.12 and similarly, Figure 4.13 is the result of parsing "Just 1".

[2] You're quite unlikely to need a prescription for aspirin but it is an example

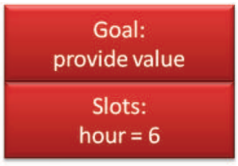


Figure 4.12: Parsing of "6 hours"

Figure 4.13: Parsing of "Just 1"

However, the previous system turns have set the expectations of the DM. Thus, if a temporary link is created between a generic class and a specific one, such as *number* → *quantity_per_intake* and *hour* → *time_between_intakes*, then the SURR may find a path from the original SFs to the expectations of the DM (Figure 4.14).

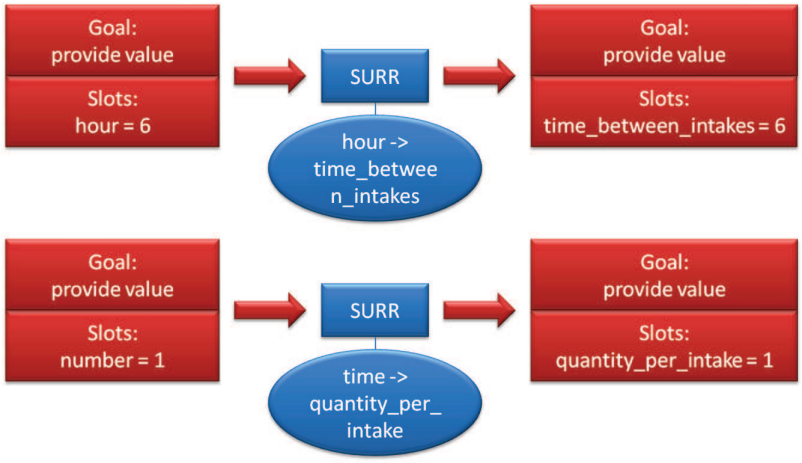


Figure 4.14: Resolution using local context

The CC intercepts the requests of the DM to create or remove branches of the SURR's forest such that the latter is always up to date with the dialog state.

4.8 Reference Resolution

Reference resolution is the process of associating absolute concepts to relative ones using the history of the interaction or any external knowledge the system can access. Two categories of references can be defined: those which refer to entities that the user assumes are known by the system and the coreferences which relate to previously debated topics.

4.8.1 Dialog Context References

We explained in previous sections (4.6 and 4.7) how the SURR, supported by the CC injects the last-turn context into the NLU processing.

Indeed, those two components capture the DM requests to update the semantic concepts trees dynamically, thus resolving ambiguities regarding shared concepts between dialogs.

4.8.2 Extended Dialog History

Currently, most of the deployed SDSs make use of the dialog history to keep track of the dialog state and disambiguate user's inputs.

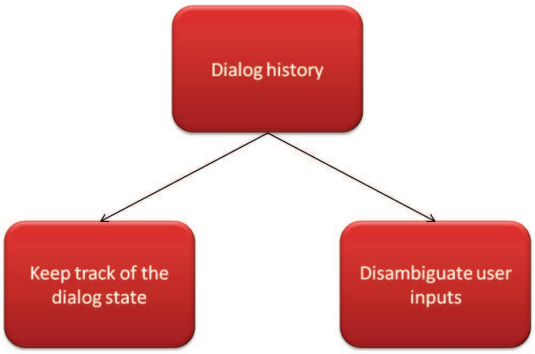


Figure 4.15: The two main usages of the dialog history

The history spans over $turn_0$ to $turn_{t-1}$. $turn_0$ is the first turn of the current dialog while $turn_{t-1}$ is the last turn, at step t, of the current interaction. For reference resolution, the user should understand that the system knows only a very restricted part of the shared knowledge, i.e. it only keeps track of the current interaction.

The NLU, includes deeper insights into the common knowledge of both entity. Indeed, we exploit the history from $turn_0$ to $turn_{t-1}$ but also the n previous turns, giving us what we call an extended dialog history which is the set $\{turn_{-n}, \ldots, turn_{-1}, turn_0, \ldots, turn_{t-1}\}$. In other words, the history is not cleared between dialogs.

In future work, we will build user's profiles from that extended history, and store and mine them efficiently for a better personalization of the SDS.

4.8.3 External References

Because the taxonomy maintained at the core of the SURR is generic, the same mechanism can be used to inject external variables into the SFs when they are relevant.

A node within the SURR includes an SF with a goal and the parameters. Some of those parameters' values are not defined but instead, when the algorithm reaches their containing nodes, trigger a call to an external resource. This source can be a database (the past weather forecast), a web-service (the weather forecast) or a set of sensors (the current local weather).

Some research teams have been studying the effects of the emotional state, the location, etc, on the way dialogs develop between a human and a machine. While this thesis did not explore these kinds of external clues to understand spoken utterances, the mechanism is available. Mostly, it has been used to read some system information for an agenda management task. Thus, the day of the week, or the current month were implicitly added to inputs such that "Monday, the 4th", for instance, was converted to an absolute concept "Monday, the 4th of October, 2014".

4.9 Semantic Unification

Semantic unification is the process of making semantic concepts converge to the same so-called semantic level.

A concept can be classified and ranked according to a taxonomy. Building taxonomies is the practice of classifying such concepts into a hierarchy.

Indeed, for the purpose of mapping a NL input to the underlying DA, it is required to get a matching between the set of concepts that can be extracted at the SP level, and the set of action parameters that can be passed on to the DM. Thus, we assume that a taxonomy classifies all our concepts and associates a semantic level to them. The problem then is to unify the semantic levels the inputs are in.

In the SURR structure, the appropriate level for the DM is the root level, e.g. the top level, although it is also possible to declare any intermediate level node as root. This, however, requires the designer to guarantee that only a single output is possible in any context. Let us consider this example (Figure 4.16) extracted from the actual taxonomy integrated in the SDS.

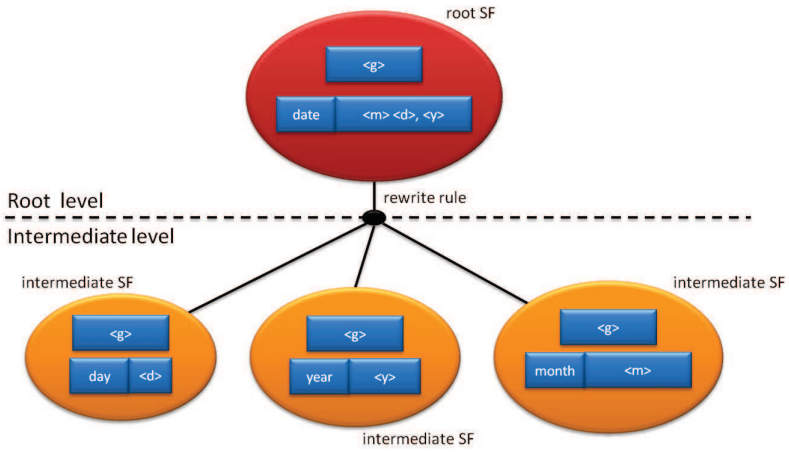


Figure 4.16: Extract from the taxonomy

Here we have 4 concepts. *date* is a particular day specified according to the Gregorian calendar, *day* is the day number of the month, *year* is the year number and *month* is the month name. Except for the concept *date*, the concepts are not root, i.e. they cannot be passed on to the DM and have to be unified to a root node. Let us further assume that a user's input got a parse like the following (Figure 4.17).

Figure 4.17: SF for the example

The SURR looking for a path to root nodes would eventually find the one represented on Figure 4.16 and thus obtain the SF in Figure 4.18 that can be passed on to the next component, thus achieving the semantic level unification task.

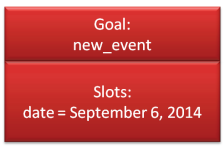


Figure 4.18: Resolution SF for the example

We remind the reader that the taxonomy can be extended as desired. More-

over in the previous example, the addition of external information such as the current month or the current year allows for the resolution of parse SFs with the *day* concept only. In such case, the *month* and the *year* concepts are instantiated with the current values. It works similarly for SFs whose only slots are the *day* and the *month* concepts.

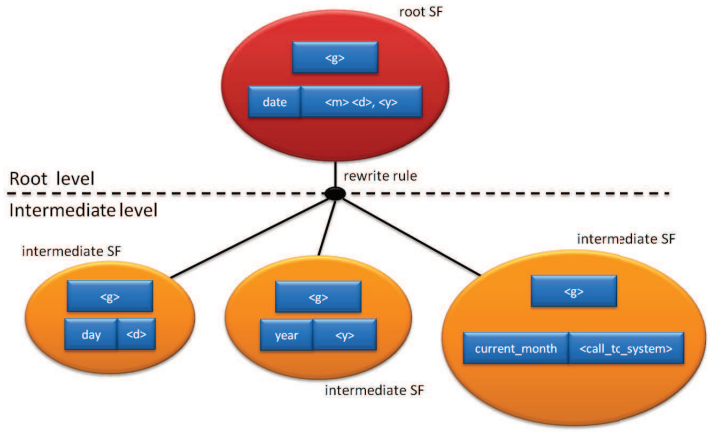


Figure 4.19: Extract from the taxonomy

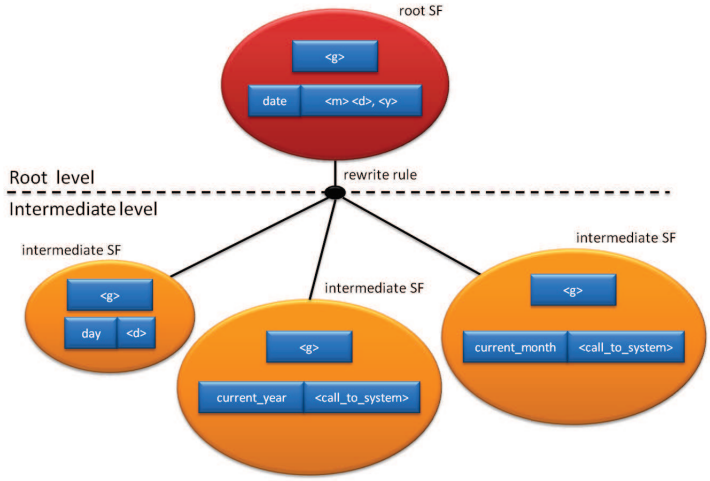


Figure 4.20: Extract from the taxonomy

The purpose of the SURR's semantic unification task brings much advantages with respect to the flexibility of the design, since it acts as an articulation between the SP and the DM. The latter's semantic space is much smaller and

much more constrained than the former's. Thus the complexity of the DM is decreased, i.e. there is no need to implement dialog models for each semantic level, while the SP tries to go as deep as possible in the parsing details, assuming that the overall meaning can be reconstructed by the SURR.

4.10 Mapping Semantic Frames to Dialog Acts

Here is the last stage of the NLU processing. Once an input has been parsed, external and local references have been resolved, and the semantic level has been unified, the ultimate step is to convert the SF into a DA.

A SF that got that far in the analysis is guaranteed to have a matching parameterized DA to trigger. However, all DAs are not available at any dialog state.

The mapper retrieves the set of DAs available when an input comes in. Then it looks for the match between the SF in and the set of DAs. This match is unique. If found, the mapper's job is to trigger the corresponding DA. Otherwise a generic "NoMap", which stands for "no mapping available", is triggered. This triggers a gradual response from the DM with a targeted guidance to the user (see section 3.8.7).

4.11 Dealing With Multiple Hypotheses

The NLU subsystem of the generic platform implements a mechanism to handle multiple hypothesis out of the speech recognizer.The method is rather simple, as schematized in Figure 4.21.

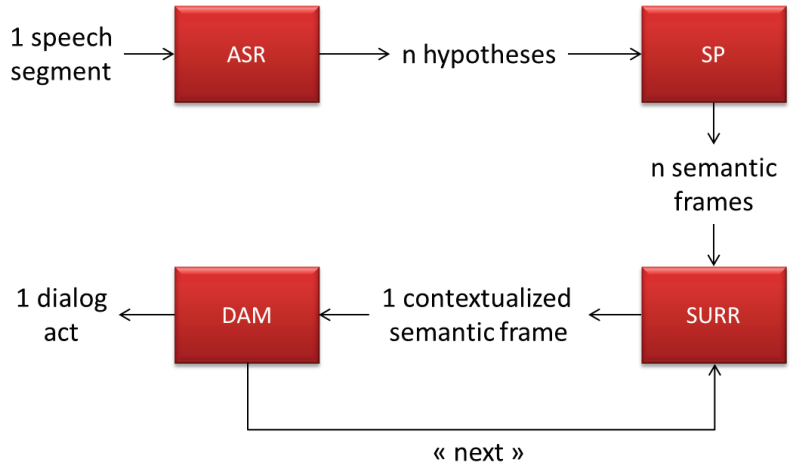


Figure 4.21: Dealing with multiple hypotheses

The ASR engine produces up to seven hypotheses from the analysis of a speech signal segment. They are ranked such that the top hypothesis is the one that got the best confidence score in the processing.

The SP extracts the meaning of every transcription, i.e. if there are seven hypotheses in, one gets seven hypotheses out of the SP module.

Then the SURR implements the storage of the hypotheses. The component first processes all of them, and then internally saves the resulting list.

A single contextualized SF enters the DAM. It is the best ranked hypothesis from the ASR, after it has been transformed by both the SP and the SURR. At that point, the final mapping of the user's turn to a DA is attempted. When it succeeds, the system's processing goes on. In the case it does not, the DAM has the ability to discard the current hypothesis and request the next one from the SURR, and this, until either the lowest ranked hypothesis is rejected or one of them is valid. If all hypotheses are rejected, the DAM bases the error recovery dialogs initialization on the best one.

The multiple hypotheses processing ability originates from the observations that many spoken utterances were transcribed with similar but different results than the expected ones. For instance, in French, "10 heures" and "Deezer", etc. Those ambiguous utterances hinders the performances of the NLU subsystem and this method solves this issue efficiently.

4.12 Evaluation

The evaluation of the NLU subsystem shows how the set of components performs when deployed to real users.

4.12.1 Corpus and method

The task of mapping natural-language utterances to situated parameterized DAs may be seen by some as a classification problem. Indeed, the overall objective is to associate a DA with the n hypotheses entering the system.

Since inter-components messages were recorded during the two real-user lab sessions, a corpus has been created, which contains 253 dialogs consisting of, on average, 3.7 turns. The total number of turns is 939.

Online and offline annotation and alignment rounds took place so that an entry, in the corpus, finally consists of the best transcription hypothesis, the SF resulting from the semantic parsing, and the DA entering the DM. Although this is not used in this evaluation, the system's DA and the generated response are included. Tables 4.8 and 4.9 show instances of turns in the database.

Best hypothesis	*à 9h du matin*
Semantic frame	*input:hour=9;*
Contextualized semantic frame	*input:first_intake=9h00*
Dialog act	*input:first_intake=9h00*
System's dialog act	*Ask.What:goal=frequency_per_day; slot=frequency_per_day;*
System's response	*Quelle est la fréquence des prises?*

Table 4.8: An example of an entry in the corpus

Best hypothesis	*mon petit-fils*
Semantic frame	*input:name=grandson;*
Contextualized semantic frame	*input:name=grandson*
Dialog act	*xxx:*
System's dialog act	*Ask.What:goal=call_type; slot=call_type;*
System's response	*Vous souhaitez passer un appel audio ou vidéo?*

Table 4.9: An example of an entry in the corpus

The SDS's NLU system processed all the hypotheses and the results were compared to the "truth" in the corpus. Several measures were taken as it will be shown next.

Metrics

Two metrics are presented, the precision:

$$precision = 100 \times \frac{count\ of\ correctly\ retrieved\ elements}{count\ of\ elements\ to\ be\ retrieved}$$

And the slot recall:

$$precision = 100 \times \frac{count\ of\ correctly\ retrieved\ slots}{count\ of\ slots\ to\ be\ retrieved}$$

4.12.2 SP Evaluation

The SP is the module which projects the text utterances into the semantic space. It has been isolated from its containing system for evaluation.

The precision of the retrieved goals is a measure of the understanding of the overall user intent, while the precision of the slots gives the variables extraction parameters.

The following chart presents the result from the measurements.

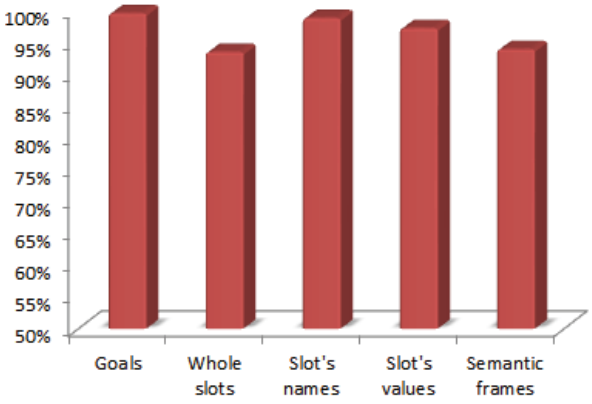


Figure 4.22: SP precision

The slot's recall is also relevant here. Indeed, as for any classification algorithm, the evaluation of the performance of the system is drawn from both the accuracy of the labeling process and the retrieving ratio. This is an estimation of the depth of the parsing whose value, in that case, is 88.27%.

4.12.3 NLU Evaluation

In a second phase of the evaluation, the same corpus was used to measure the performance of the whole NLU system. Figure 4.23 shows the setup of the experiment.

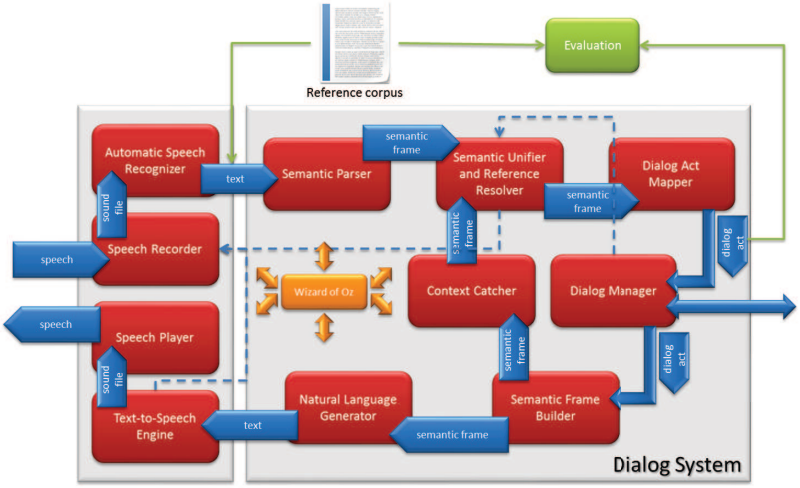


Figure 4.23: NLU evaluation setup

The count of DAs that matched the ones contained in the reference corpus were 912, which means that, out of 939 text utterances, 97.12% have been fully understood. 2.77% of the resulting DA were correct but they lacked some information and 0.11% of these, i.e. only one instance, contained errors.

4.13 Conclusion

In this chapter, the NLU subsystem has been presented. It consists of a network of independent services implementing non-overlapping tasks. The inputs are textual hypotheses based on the user's utterances' content and the outputs are contextualized parametric DAs. The NLU is the link between the ASR module and the DM core.

The architecture and the specifications of the NLU part have been conceived such as to be dialog independent. This includes the way the interaction is managed and the domain it belongs to. Such generic components allow the actual implementation of a wide range of SDSs for many applications.

The performances of the system has been evaluated through user trials. Results show that the system works efficiently. These measures depend on the amount of available data and the interaction variability.

Since every individual service in the NLU set is independent, replacing them is straightforward. Substituting a component with a WoZ operator, for data collection or comparative evaluation, is also easily possible.

Finally, the simple task carried out by each part of the subsystem requires from developers only little knowledge about the underlying SDS technology, allowing them to quickly set up and configure a system. Such, also helps to open the domain to a wider community of potential SDS designers.

Still aiming at facilitating and spreading the implementation of SDSs to interface various applications,the last stage of input processing takes place within the DM. A tool has been proposed to ease that task.

Chapter 5

The Linked-form Filling language: A New Paradigm to Create and Update Task-based Dialog Models

5.1 Introduction

The ultimate goal of the artificial intelligence is to emulate the human brain abilities. In 1950, Alan Turing asked: "Can a machine think?", the famous Turing test was born. The competition asks several human testers to engage in a conversation with a software, which is either based on an algorithm (i.e. some sort of artificial intelligence) or operated by a human being. The participants are then asked to estimate the likelihood they would assign to the system being operated by a human.

According to the research community, the first so-called DM was named ELIZA [204]. Its best script, according to most, was the one where she (it) mimicked the behavior of a psychotherapist. It was based on adjacency pairs, i.e. the system reacted to keywords and patterns, spotted in the textual entries, with predefined replies.

A DM maintains a dialog context, processes the user's inputs and questions the interaction models to select the most appropriate next step in the dialog [82]. Managing dialogs is necessary when the length of the interaction exceeds one turn (cf. 2.1).

In this chapter, we therefore introduce a new coding language to support the design of ANSI/CEA-2018-compliant [41, 161] dialog models. After reviewing the state of the art in dialog management, the task hierarchy paradigm is introduced. Then examples of models are given. Based on the conclusion drawn from experimenting the design of such models, an alternative is proposed, its

trade-offs are discussed, and some advantages of using this LFF paradigm for dialog modeling are presented.

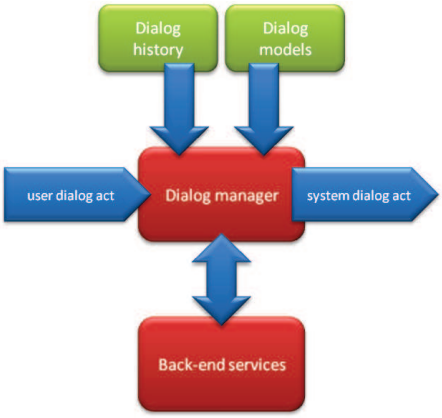


Figure 5.1: The DM task

5.2 Related Work in Dialog Management

Following is an overview of the history of various dialog management methods [29, 37, 88, 107].

5.2.1 Flow Graphs

Definition A flow graph is a deterministic representation of the different paths a program can go through while executing. In the dialog management domain, the nodes of such graphs contain the dialog state and the action the system performs. Directed edges are the transitions from one such state to another. They are conditioned on the user's turns.

Example Figure 5.2 shows a possible flow graph for a messaging application.

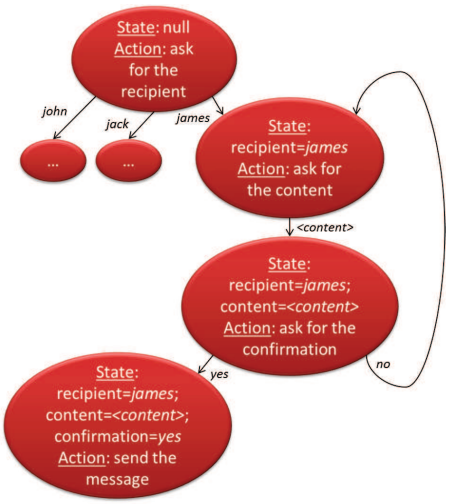


Figure 5.2: Flow graph example

Advantages and drawbacks Designing flow graphs to model dialogs is fairly easy and straightforward. However, the depth of the interaction, its apparent intelligence and its flexibility, are rather limited.

5.2.2 Adjacency Pairs

The adjacency-pairs paradigm is a scripted dialog management method acting on a local level. Adjacency pairs, as used by ELIZA [204], rely on the detection of keywords and patterns in the user's inputs to trigger predefined replies.

This is well suited to build shallow automatic conversational partners since the system does not explicitly maintain a state of the dialog, and neither keeps track of the history.

Moreover, there is no target goal to achieve but, instead, the available rules maintain a coherent dialog in the topic of the user's choice. Chatterbots are mostly based on this rule formalism.

Examples of adjacency pairs:

Pattern	Predefined reply
"What's your name?"	*My name is Eliza*
"[...]mother[...]"	*Tell me about your mother*
...	...

5.2.3 The Information State

The theory behind the IS [101, 134] approach builds on the idea that the state of a dialog is based on the informative content exchanged while interacting. Track-

ing the IS means monitoring, storing and reasoning based on the information cumulated over the course of a dialog session.

Setting up an IS manager requires the execution of a sequence of steps.

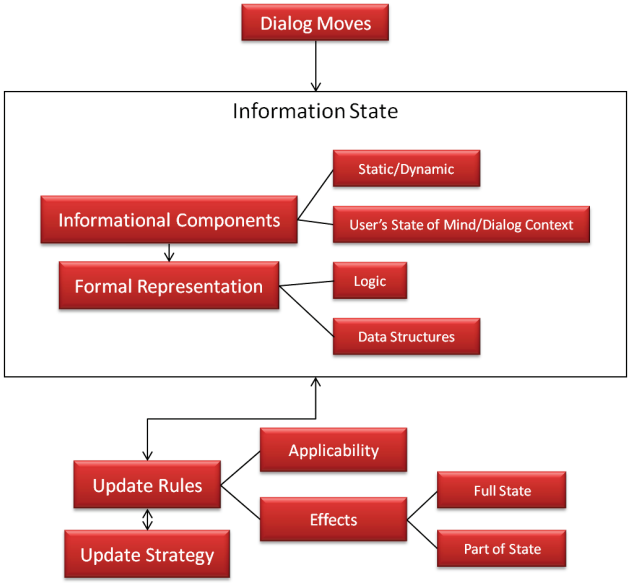


Figure 5.3: Scheme of the information state process

Identifying the key informational elements of the dialog As shown in Figure 5.3, the basic elements of the IS are the informational components. These are the bits and pieces of the information that the DM keeps track of. They can be qualified as static, dynamic, semi-static/semi-dynamic according to the granularity of the information components.

A static piece of information is available anytime and cannot be modified by the interaction. The name of the system's avatar is, for instance, a static information.

The current date or time are semi-static/semi-dynamic since they are not modifiable with dialogs while they change over time.

As for the dynamic pieces of information, the current intent of the user, the last dialog move and the system's objective are instances of those.

The informational components may represent the user's state of mind (as observed by the system), the system's dialog context or the shared knowledge between participants.

Defining the set of dialog moves Dialog moves are the dialog action units. They are the result of the processing of the user's signal by the interpretation components of the dialog system. The IS designer defines such a set of dialog moves that would trigger some update rules.

Providing an update mechanism The IS evolves following update rules. They are made of two parts. The applicability conditions analyzes the IS and the last dialog move, i.e. the last user's DA, to allow or prevent the execution of the rule's operation. If the applicability is valid, the effects can be applied.

Effects update the IS (if necessary) partly or fully and provide it with the next system's dialog moves.

Providing an update strategy The last component in an IS system is a set of update strategies. Indeed, there is no requirement to insure that only a single update rule is applicable for every pair *(state, dialog move)* . Thus, it is necessary to have a method to select the rule that will operate in cases where several rules are applicable.

Update strategies range from random selection to more elaborated ranking mechanisms, to reduce a set of several update rules to a single one.

The IS approach is a theory with little guidances and no/barely any restrictions with respect to the low-level implementation. Moreover there is no necessary finiteness in the dialog, i.e. the IS can grow indefinitely. Just like the scripted adjacency-pairs paradigm, the IS is suited for question-answering characters and has been demonstrated as such.

5.2.4 Example-based Dialog Modeling

As highlighted before, the main drawback of dialogs based on rules is that their conception is very human-labor intensive. Lee et al., however, proposed a hybrid system which is based on rules [91, 106, 108, 109], while trained from annotated data. They followed two principles. One is that the DM task should be free of state transitions. The other is that the domain specificities of a task should be processed by experts of that domain.

In order to follow these incentives, the example-based DM does not base its reasoning on finite-state models but instead deals with the "situation". A situation consists of the current user utterance, the user intention, the semantic frame[1] and the discourse history.

User utterance	"I want to go to Dublin at 7 pm"	
User intention	Dialog act	request
	Main action	travel_booking
Semantic frame	destination = Dublin; time = 7 pm"	
Discourse history	date = September 6, 2014; departure = unknown; destination = unknown; time = unknown	

Table 5.1: An example of "situation"

[1]Note that here the concept of an SF is different from an SF introduced in Section 4.2.6

This closely relates to the IS. A situation is the summary of all information elements that matter to reason and select the next step to take in the dialog.

The DM rules are learned from an annotated corpus of dialogs. The annotations match the requirements to build situations. They combine the text utterance, the intention and the SF as extracted by the NLU. These annotations can be obtained from an annotated corpus of the NLU/NLG system or by running those algorithms on a raw corpus. Each turn is an entry of the example database to which the discourse history is associated.

Figure 5.4 shows the schematic of the DM processing.

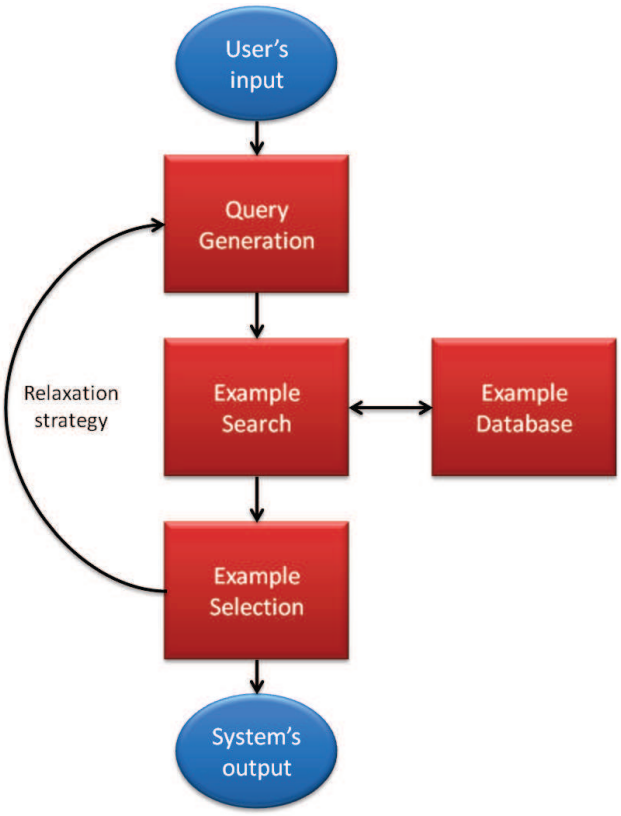


Figure 5.4: Cycle of an example-based dialog management process

The input of the user, augmented with the output of the NLU, is transformed into a query for the database of examples. Then, there is either no match, a single one, or multiple matches found. Depending on this result, the DM, respectively, relaxes some constraints of the query, produces the system turn of the example retrieved from the database or proceeds to a deeper selection among the results.

The relaxation strategy operates on the DA and on the main action of the situation to broaden the search. The selection mechanism computes a similarity measure based on lexico-semantic and historical criteria in order to select a single output turn.

5.2.5 Markov Decision Processes

Definition The dialog management task can be cast as a finite-state process and modeled by an MDP. A MDP [113, 114, 147, 167], applied to dialog management, is defined by the tuple $\{S, A, T, R, s_0, \pi\}$ where:

- $S = \{s_0, s_1, ..., s_m\}$ is a set of states
- $A = \{a_0, a_1, ..., a_n\}$ is a set of actions
- $T : S \times A \times S \rightarrow \mathbb{R}$ is a transition probability function
- $R : S \times A \rightarrow \mathbb{R}$ is a reward function
- s_0 is the initial state
- π is a policy

The states s_i are instances of the configuration the dialog may be in. In the case of a booking application, for instance, the states can enclose the departure location, the destination, the time of departure, etc. The choice of the variables to represent the dialog state is of great importance. All the information relevant to the action selection have to be included. The value of the variables is maintained in the states as well.

The actions are the ones that the system may take, such as requesting an input, informing the user, greeting the user, etc.

Transiting from one state to another has a cost defined by the probability function T. $T(s, a, s') = P(s'|a, s)$ is the probability of transiting from state s to state s' taking action a.

Finally, the designer of such models is required to implement a reward/cost function which evaluates the efficiency of taking an action a while in state s. This function rewards the "good" decisions of the system and the achievement of the task to be implemented.

The policy π, which is learned from data, associates each state with the action to take when in it. This is not part of the initial human-created model, the policy is trained. When running the model, it substitutes the reward function to control the flow of the dialog.

Execution The execution of MDP models follows these steps:

1. while in state s_0, select an action a_0 according to policy π
2. receive a reward $R(s_0, a_0)$

3. move to $s_1 = \arg\max_{s'} T(s_0, a_0, s')$
4. while in state s_1, select an action a_1 according to π
5. receive reward $R(s_1, a_1)$
6. move to $s_2 = \arg\max_{s'} T(s_1, a_1, s')$
7. ...
8. move to s_t, a terminal state

Training The training algorithm looks for the policy that maximizes the cumulative reward for the training corpus. This is given by:

$$\sum_{t=0}^{\infty} R(s_t, a_t)$$

It is is based on the value function which is the value of the expected reward starting from state s and applying the policy π. That is:

$$V_\pi(s) = R(s, \pi(s)) + \sum_{s \in S} T(s, \pi(s), s') \cdot V_\pi(s')$$

5.2.6 Partially Observable Markov Decision Processes

Partially Observable Markov Decision Processes (POMDPs) extend from MDPs to include uncertainty in the decision-making mechanism [25, 28, 64, 79, 81, 94, 119, 135, 148, 149, 150, 190, 191, 192, 193, 206, 207, 208, 209, 210, 216, 217, 218, 221]. The dialog states are hidden and can only be estimated with the generated observations.

Definition The components of a POMDP are in the tuple $\{S, A, T, R, O, Z, b_0, \gamma, \pi\}$ where:

- $S = \{s_0, s_1, ..., s_m\}$ is a set of states
- $A = \{a_0, a_1, ..., a_n\}$ is a set of actions
- $O = \{o_1, o_2, ..., o_o\}$ is a set of observations (may be a continuous space)
- $T : S \times A \times S \to \mathbb{R}$ is a transition probability function
- $R : S \times A \to \mathbb{R}$ is a reward function
- $Z : S \times A \times O \to \mathbb{R}$ is an observation function
- b_0 is an initial distribution over the state occupancy
- γ is a geometric discount factor

- π is a policy

S, A, T and R are similar to the spaces and functions of an MDP.

O, the set of observations or observation space, includes all vectors the system can get from the user, i.e. those are the user actions.

$Z(s', a, o) = P(o|a, s')$ is the probability of observing o after taking the action a and transitioning to state s'.

Since the system's state is not known for sure due to the uncertainty over the user's input, tracking it is achieved by computing, at each time step, a distribution of the probability of being in each one of them. This dynamic distribution is called the belief state b. $b_t(s)$ is the probability of being in state s at time t. Initially, the belief state b_0 is arbitrarily defined and in most cases, $b_0(s)$ is null for all but one state which is, by analogy to the MDPs, s_0.

Training Tracking the belief state means computing the new belief state b' at each step based on the previous belief state b, the last action selected a, and the observation received o.

$$b'(s') = k \cdot Z(s', a, o) \sum_{s \in S} T(s, a, s')b(s)$$

k is a normalizing constant: $k = 1/P(o|b, a)$

The cumulative reward is then:

$$\sum_{t=0}^{\infty} \gamma^t R(b_t, a_t) = \sum_{t=0}^{\infty} \gamma^t \sum_s b_t(s) R(s, a_t)$$

The value function, that gives the expected reward starting from the belief state b and following the policy π is computed as follow. We write:

- $b' = \tau(b, a, o)$ the belief updating function
- $R(b, a) = \sum_s b(s)R(s, a)$ the reward function
- $Z(b, a, o) = P(o|b, a)$ the observation probability

The value function for training becomes:

$$V_\pi(b) = R(b, \pi(b)) + \sum_{o \in O} Z(b, \pi(b), o) \cdot V_\pi(\tau(b, \pi(b), o))$$

5.3 The Task Hierarchy Paradigm

There is one DM method that has not been mentioned in the above section. It is the task hierarchy paradigm.

5.3.1 Principles

Modeling dialogs may be achieved by splitting them into fine-grained tasks whenever it is possible [15, 18, 20, 72, 112, 161, 162, 163, 164, 165, 182]. The ruling theory is that a task is completed when all its subtasks are. A primitive task, which has no subtask, is achievable by itself.

The conception work of a task model consists in defining unit tasks and organizing them onto a hierarchy such that a dialog is a node in the hierarchy, which requires all of its children tasks to be executed.

5.3.2 The ANSI/CEA-218 Standard

The ANSI/CEA-2018 standard is a framework for task model descriptions, i.e. it defines semantics and Extensible Markup Language (XML) notations for them. The Disco dialog management engine makes use of such models to manage dialogs.

The ANSI/CEA-2018 standard [41] was originally intended to provide a unified intermediate layer between electronic devices and GUIs. The functionalities of the standard are shown in Figure 5.5.

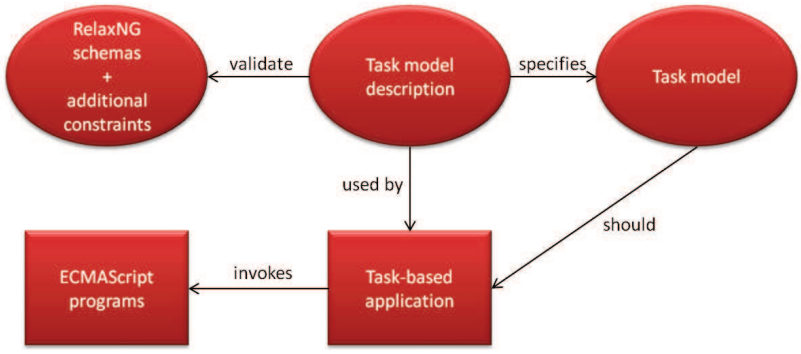


Figure 5.5: ANSI/CEA-2018 interaction scheme (source: [41])

A dialog model, or task model description, is a tree structure defining a hierarchy of tasks with their attributes: input and output parameters, pre- and post conditions, bindings, task decompositions, grounding scripts, etc.

The top task is an XML element which links to sub-nodes, which have to be completed to achieve a dialog. The root of any task file is a <TaskModel> element.

Theory

The ANSI/CEA-218 Standard is a framework to create task model descriptions. The units of such models are task classes which have several parameters characterizing each one of them: the time extent, the task actor – the entity performing

the task –, the abstraction level, etc. The same task class can be instantiated at will.

In the next subsections, the XML elements defined by the standard are described.

Syntax

taskModel The root node of every task model. The content of the "about" attribute defines a (unique) namespace associated with the model.

task A task may have a precondition, a postcondition, a set of subtasks, a set of scripts, a set of inputs and a set of outputs.

$$\text{<task>} = \begin{array}{l}(\text{<precondition>})? \\ (\text{<postcondition>})? \\ (\text{<input>})* \\ (\text{<output>})* \\ (\text{<subtasks>})* \\ (\text{<script>})*\end{array}$$

Tasks are the basic entities of task models.

input and output A task may require variables to be defined or may create/-modify some while executing. Such is the motivation of the input and output nodes. The former is a set of variables that need to be given a value before the task containing them can be initiated. The outputs are accessed as the task is executed. They are bound to input variables and/or handled by ECMAScript scripts [58].

Four of those slots are predefined. The input slot whose name is external, of boolean type, defines the actor for the task, which is either the user (external = "true") or the system (external = "false"). The slots "device", "when" and "success" are not relevant here.

precondition and postcondition The execution/success of a task may be parameterized by the return value of an ECMAScript script. A precondition and a postcondition contain scripts which return boolean values to allow the execution of a task and mark it as successful respectively.

subtasks A subtasks node describes a method, i.e. a set of steps, to achieve a task. It defines sub elements, which share the same precondition, postcondition, inputs and outputs. A subtasks node consists in a recipe to achieve the task it belongs to.

$$\text{<subtasks>} = \begin{array}{l}(\text{<step>})* \\ (\text{<applicable>})? \\ (\text{<binding>})*\end{array}$$

step A step is a link to a task. The modularity of the standard allows for tasks to be inserted into different subforms by reference. Moreover, a step links to a task according to some conditions. The "minOccurs" and "maxOccurs" attributes define the range of the task's occurrence – it is higher than "minOccurs" and lower than "maxOccurs". The "requires" attribute points to a list of steps that have to be achieved before the containing one can be entered.

applicable The applicability condition of a subtasks node can be explicitly specified with an ECMAScript script that returns true (the subtasks are applicable) or false (the subtasks are not applicable)

binding Within a binding element, the variable pointed by the slot's attribute is set to the value of the value's attribute. Bindings unify local variables throughout tasks. The first attribute's value is set to the "value" attribute's content interpretation.

script Grounding steps of the dialog are implemented in the ECMAScript language embedded into script elements that can be called by tasks.

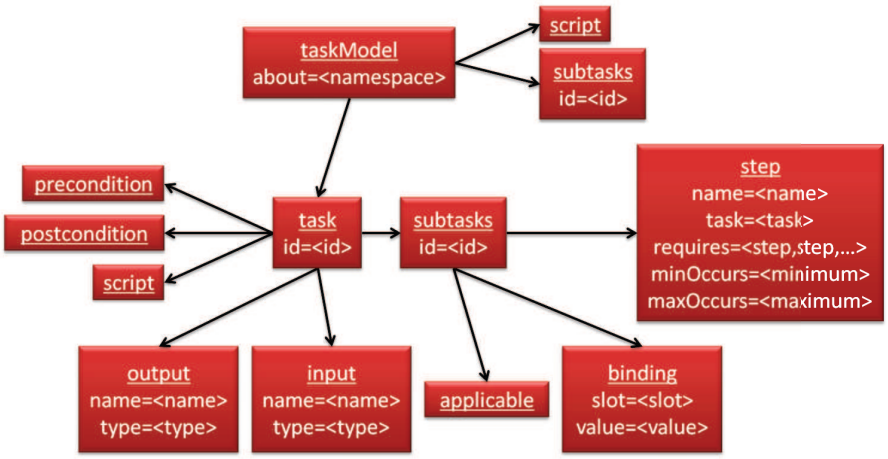


Figure 5.6: The ANSI/CEA-2018 schematic (some details have been omitted)

5.3.3 Related Issues

Simple task model

Let us consider, as an example, a possible dialog model to send a message to someone. The root task is decomposed into several primitive ones. Figure 5.7 shows the task decomposition. The "set_content" and the "set_recipient" tasks require, respectively, the text content of the message and the recipient's name to be defined before executing.

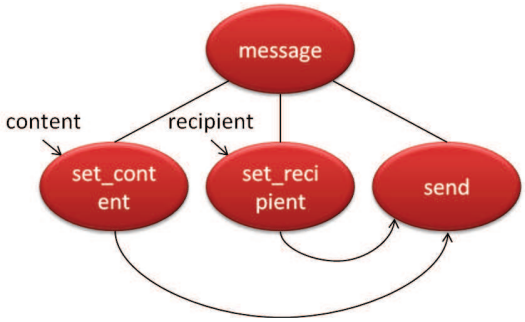


Figure 5.7: An example of a simple task model.

Following is the code implementing the whole task model. Note the addition of an attribute "requires", which forces both tasks set_content and set_recipient to be executed before the "send" task can be started. This attribute is available because the "ordered" boolean is set to false (default is true).

```
<taskModel>
   <task id="message">
      <subtasks id="message_steps" ordered="false">
         <step name="set_content" task="set_content"/>
         <step name="set_recipient" task="set_recipient"/>
         <step name="send" task="send" requires="set_content set_recipient"/>
      </subtasks>
   </task>
   <task id="set_content">
      <input name="content" type="string"/>
      <binding slot="$this.external" value="false"/>
      <script><!– grounding script to set the content –>
      </script>
   </task>
   <task id="set_recipient">
      <input name="recipient" type="string"/>
      <binding slot="$this.external" value="false"/>
      <script><!– grounding script to set the recipient –>
      </script>
   </task>
   <task id="send">
      <binding slot="$this.external" value="false"/>
      <script><!– grounding script to send the message –>
      </script>
   </task>
</taskModel>
```

Table 5.2: An example of a simple task model XML code.

The example is quite shallow thus simple to understand, debug or extend.

Instantiating this messaging task model description, a typical dialog would be:

User U1: *I'd like to write a message*	[Propose to start the message task]
System S1: *Who would you like to send it to?*	[Ask for the recipient input for set_recipient]
User U2: *To John*	[Propose a value for the recipient input for set_recipient]
System S2: *What would you like to write?*	[Ask for the content input for set_content]
User U3: *The meeting is at 5*	[Propose a value for the content input for set_content]
System S3: *It's done*	[Close the dialog]

Actually, if the surface form of the turns is not considered, there are only two sequences of task execution. The other one is:

User U1: *I'd like to write a message*	[Propose to start the message task]
System S1: *What would you like to write?*	[Ask for the content input for set_content]
User U2: *The meeting is at 5*	[Propose a value for the content input for set_content]
System S2: *Who would you like to send it to?*	[Ask for the recipient input for set_recipient]
User U3: *To John*	[Propose a value for the recipient input for set_recipient]
System S3: *It's done*	[Close the dialog]

An experienced user would, however, like to get rid of some guidances and thus be able to skip some tasks or, in other words, merge them. For instance, User U1 and User U2 may be concatenated into a single utterance: *I'd like to write a message to John.* The same goes for larger models, where the user should be able to fill as much fields as he/she would like. Including those dialog alternatives in the models expands them.

Complex task model

Let us now consider a larger model. One which is even too complex and large to be drawn in this document despite a fairly simple dialog basis. Here is a description of the dialog model in English words. It is extracted from the set of models built along the course of this thesis and was generated using the Extensible Stylesheet Language Transformations (XSLT) rules described later (Section 5.6), i.e. it was not hand coded.

"add_a_prescription" dialog model The application is a personal drug management software that basically organizes one's medical prescriptions. The

user adds new prescriptions via a spoken dialog. In Table 5.3 the mandatory fields that need to be filled are detailed for the sake of clarity.

Field's name	**Details**
drug	Name of the medication
frequency_per_day	Frequency of intake per day
quantity_per_intake	Quantity of medication unit per intake
dosage_form	Medication unit form
interval_time	Time interval between intakes
first_intake	Time of the first intake of the day
confirmation	Validation confirmation
field	Selection of the field to be changed

Table 5.3: Field description.

When sufficient data has been collected by the system, a summary is displayed to the user in order for him to validate the prescription or change the incorrect field(s).

Additionally there is another source of increased complexity: the synchronization with the GUI. Every field the SDS gets a value for is stored in memory as well as displayed on the screen. This is achieved through grounding scripts that send the update to a message bus. These scripts need to be added at every relevant point of the dialog flow.

Issues with complex models The "add_a_prescription" dialog task model defines 42 tasks and one initial script. A designer of such task models, needs a way to design a set of interactions, define the minimal constraints of the dialog and let the system mine that to generate every possible turn sequence.

The advantages are two-fold. A modification of the dialog structure is automatically echoed throughout the whole task model. The system itself browses the design to construct a model including all the possible paths to achieve dialogs.

Aiming at these two features, both a description language and a set of recursive XSLT rules to convert from documents in the new language to similar models compliant to the ANSI/CEA-2018 standard, have been proposed.

5.4 Disco: A Dialog Management Library

The core of the implemented DM is based on Disco [72, 112, 161, 162, 163, 164, 165, 182], an open-source dialog management library, whose algorithm processes task hierarchy models.

5.4.1 Introduction

A dialog model is a constrained XML tree of tasks [161]. A terminal task node, also called an action, is executable with no other purpose than itself, whereas the other tasks are decomposed into subtasks.

In Disco, an internal stack maintains the state of the dialog. It piles up tasks, associated with partially elaborated plans, to be completed to achieve the current dialog(s).

The plan recognizer uses the recipes defined in the dialog models and this dialog state to select the best available plans for the tasks in the stack. Then the reasoning engine selects the most appropriate next step.

5.4.2 Embedding Disco

According to Disco's basic theory, a dialog event may have three effects. It either:

1. starts a new discourse segment
2. contributes to the current discourse segment, or
3. ends the current discourse segment

This translates into a set of DAs which either:

1. start a new discourse segment
2. execute a step in a recipe
3. identify a recipe
4. identify the actor of a task, or
5. identify the input parameter(s) of a task

A mechanism had to be implemented to map the SFs out of the NLU component with Disco's commands. This process is dynamic because a DA modifies the internal state of the DM. The set of available commands applicable to a dialog context is finite. The mapping algorithm is a multiple-stage process (cf. 4.10) that is schematically described in Figure 5.8. It returns an ordered list of DAs that are sequentially fed to the dialog engine.

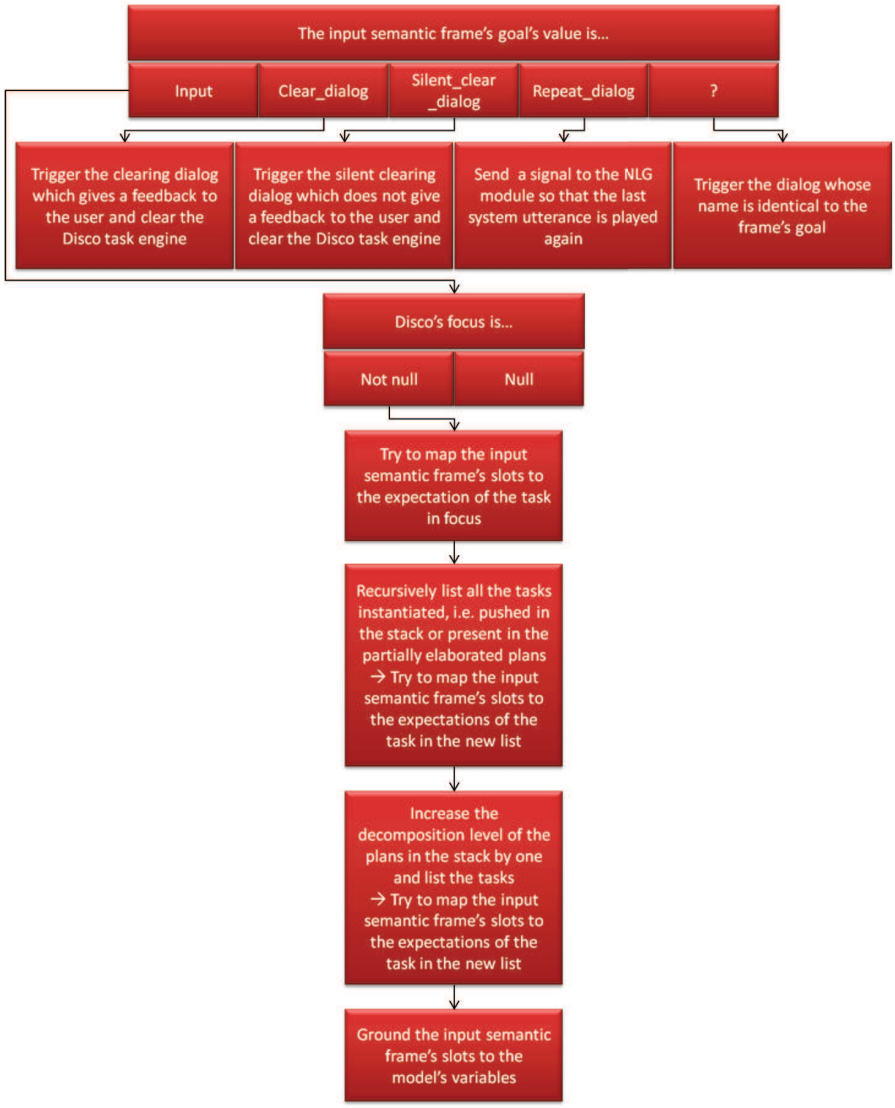


Figure 5.8: Schematic of the algorithm mapping SFs to DAs

On the output side of the DM, another interface is required to convert the agent's utterance format to SFs.

Disco's creators proposed a set of XSLT rules to facilitate the design of model files [164]. However, one may quickly find himself overwhelmed while building, debugging and maintaining even small-sized models, due to both the lack of an appropriate GUI interface and the complexity and depth of the models.

5.5 Linked-form Filling Language Description

First, in order to automatize the building of dialog models, application specifications were described in terms of logical operations between variables. Then, the LFF has been proposed.

5.5.1 LFF Principles

In an attempt to overcome the hurdles inherent to the specification of task models, the dialog modeling paradigm is shifted to an LFF one. Form-filling dialogs are based on structures containing sets of fields, which the user needs to provide a value for in order to trigger a terminal action. The order in which the DM asks for the values is not predefined. The user may define multiple field values within a single utterance/turn.

The LFF language offers to combine these properties with the ability to trigger an action at any point of the dialogs and the inclusion of subforms. Furthermore, a field, just like a subform, can be optional, i.e. either ignored by the system when unset or proposed to the user.

Let us get back to the simple dialog example described previously (see section 5.3.3). The form-filling of such a dialog is shown in Figure 5.9.

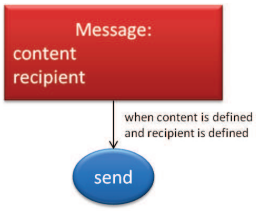


Figure 5.9: Form-filling example 1

A user cannot confirm that the message content is correct before sending it. This is solved by:

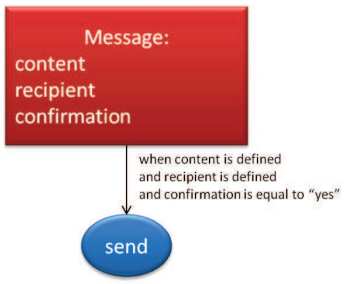


Figure 5.10: Form-filling example 2

We wish to execute some actions at key points of the dialog. The issue here is that if the order of the slots is not constrained, we may end up with the "confirmation" field defined before the "content" one, resulting in an invalid dialog state. Furthermore, we would have to define every possible combination of slots statuses to trigger the right action after filling one of them (cf. Figure 5.11).

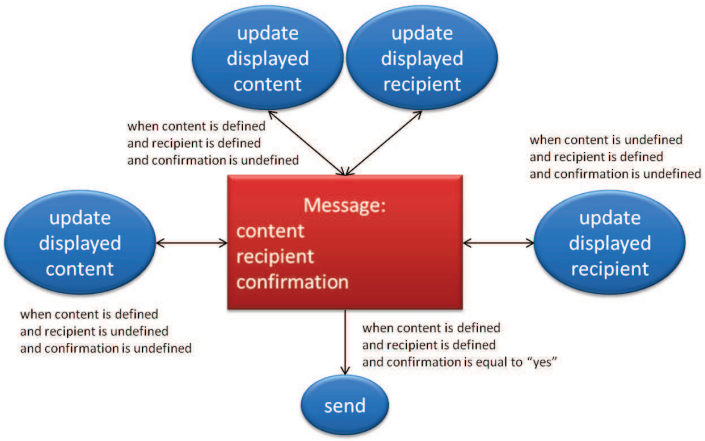


Figure 5.11: Form-filling example 3

Forcing a strict order is also not a good solution, as this would prevent the redefinition of incorrect data. Here, we make use of the unlimited depth of a task model – because a link between two task nodes is a reference, a node can point to its "parent" node – to loop around tasks while keeping a sequencing order. Figure 5.12 is the schematic of the message dialog in this new paradigm. The numbering of the arrow is an indication of the linking sequence order.

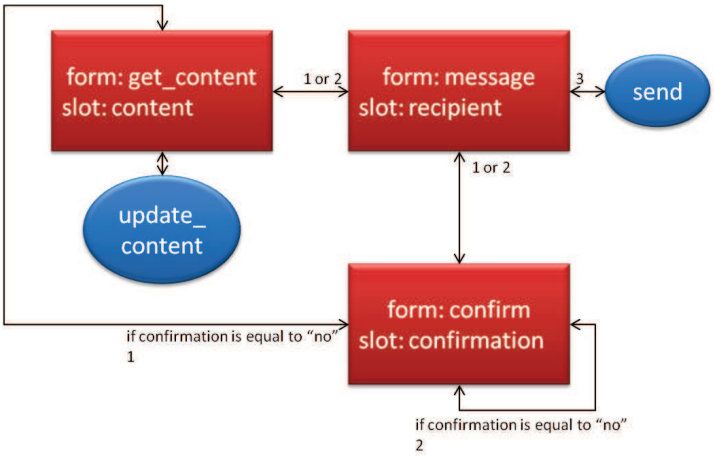


Figure 5.12: LFF language description example

The reader may notice the two links labeled "If confirmation is equal to "no"". Those are conditional links which are dependent of the value of the "confirmation" slot, i.e. they are both applied or none of them is.

It is encoded via the following lines

```
<description>
    <form id="message">
        <slot variable="recipient"/>
        <subform constraint="mandatory">
            <link form="get_content"/>
            <link form="confirm"/>
        </subform>
        <subform constraint="mandatory">
            <script action="send"/>
        </subform>
    </form>
    <form id="get_content">
        <slot variable="content"/>
        <script action="update_content"/>
    </form>
    <form id="confirm">
        <slot variable="confirmation"/>
        <subform constraint="mandatory">
            <link form="get_content"/>
            <link form="confirm"/>
            <applicable>
                confirmation=="no"
            </applicable>
        </subform>
    </form>
    <variable id="recipient" type="string"/>
    <variable id="content" type="string"/>
    <variable id="confirmation" type="string"/>
    <action id="send"/>
    <action id="update_content"/>
</description>
```

Table 5.4: LFF language description XML code.

5.5.2 Syntax

Following is the syntax of the LFF language.

description The root node of all LFF (LFF) documents is a description element

form Forms are the main entities to be defined in the language. They are tagged with a unique identifier (ID). Links reference these IDs.

action The content of an action is an ECMAScript script that can be called by a script node, pointing to the action's ID. Here, the ID also needs to be unique.

variable Variables are convenience nodes. Each one refers to a global variable. A smarter way of dealing with global declarations would have been to list all the slots of the model and declare a unique instance of each one.

subform Subforms are contained within forms. They themselves contain a set of zero or more links to either forms (<link> elements) or actions (<script> elements)

slot A slot is the unit variable in a form. In other words, a slot needs to be given a value to complete the form.

link Links refer to forms. It is basically the insertion of a form into the containing subform.

applicable Similar to the applicable nodes of the ANSI/CEA-2018 standard. A script returns a boolean value to allow or disallow the execution of the parent subform.

script Self-named scripts are grounding actions to apply while executing a subform.

Figure 5.13 shows the hierarchy of XML nodes, as specified by the new LFF language.

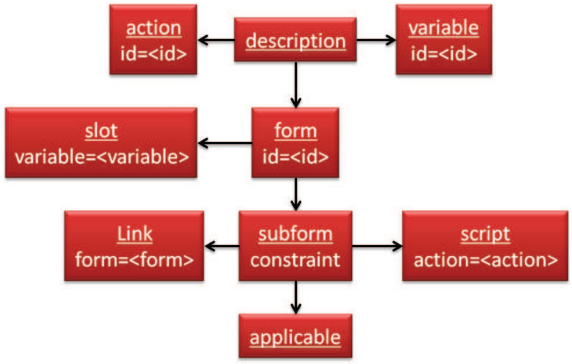


Figure 5.13: Hierarchy of the LFF language

5.6 From Linked-form Filling to ANSI/CEA-2018

The aim of the LFF language is to offer a somehow simpler design method to a powerful standard dialog modeling specification. Since it is also an XML based language we opted for XSLT to convert an LFF document into a compliant dialog model. XSLT stands for EXtensible Stylesheet Language Transformations. A number of rules have been defined to create a well-formed LFF document. This section will not go through all of them but will describe the basic principles of the transformations.

5.6.1 Variables and Actions

Variables and actions generate primitive tasks identified by their ID. They are executed by the system. Below is an example of transformation of a variable element toward a primitive task.

LFF	<variable id="variable_id"/>
ANSI/CEA-2018	<task id="variable_i"> <input name="variable_id" type="string"/> <binding slot="$this.external" value="false"/> <script>variable_id=$this.variable_id; </script> </task>

Table 5.5: Transformation of a variable node

Here is the same transformation type applied to an action node

LFF	<action id="action_id"> ECMAScript </action>
ANSI/CEA-2018	<task id="action_id"> <binding slot="$this.external" value="false"/> <script> ECMAScript </script> </task>

Table 5.6: Transformation of an action node

5.6.2 Forms

A form is transformed into a task with the same ID. The rules applied to a newly created node are split into two parts.

The first set of rules recursively creates the inputs based on the slots of the current form, but also all the descendant slots defined by the linked forms. The subtasks nodes are created in three steps: the step nodes, the applicable node, and the binding nodes.

The transformations are applied to maintain the constraints defined at the LFF level so that a subform whose constraint is "ignored" does not transform the same way as one whose constraint is "mandatory" or "proposed". Also, all the possible dialog paths are explored to make them available in the task tree.

Following is an example of the conversion of a form element. The example is incomplete since the slot element makes a reference to a non-existing variable. The link and the script tags are void as well.

LFF	<form id="form_id"> <slot variable="variable_id"/> <subform constraint="mandatory"> <link form="form_id_2"/> <script action="action_id"/> </subform> </form>
ANSI/CEA-2018	<task id="form_id"> <input name="variable_id" type="string"/> <subtasks id="form_idSteps1"> <step name="empty" task="empty"/> <step name="variable_id" task="variable_id"/> <step name="form_id_2" task="form_id_2"/> <step name="script_action_id" task="action_id"/> <applicable>($this.variable_id!="null") </applicable> <binding slot="$variable_id.variable_id" value="$this.variable_id"/> </subtasks> <subtasks id="form_idSteps0"> <step name="empty" task="empty"/> <step name="variable_id" task="variable_id"/> <step name="form_id_2" task="form_id_2"/> <step name="script_action_id" task="action_id"/> <applicable>($this.variable_id=="null") </applicable> </subtasks> </task>

Table 5.7: Transformation of a form node

5.7 Linked-form Filling Evaluation

The evaluation of a new programming language to model dialogs deals with two questions:

1. How does the new models compare to the former ones at runtime?
2. What are the advantages and drawbacks of using the new paradigm for a designer?

Both directions are explored in the following sections.

5.7.1 Model's Comparison

In order for a DM such as Disco to run them, LFF models are "compiled", i.e. converted using XSLT rules. The result of the transformations is a task-

based model compliant to the ANSI/CEA-2018 standard. This means that the final models are actually identical, whether they have been manually created or derived from LFF intermediate models.

The LFF does not add any functionalities nor any mechanisms to dialog models. It is a supporting tool to create them. The quality of a model depends mainly on the skills of the designer who implements the standard directly. While it would surely take more time to create the models, one may indeed avoid using the LFF.

5.7.2 Design Comparison

Size and complexity measures

All the dialog models available by the time of writing this thesis – 26 of them – have been evaluated. A human designer created the LFF models and the automatic tool performed the transformation to ANSI/CEA-2018 compliant structures. Two comparisons measuring the design complexity are shown in Table 5.8.

	Task hierarchy	LFF	Reduction
Count of lines	139 $[49 - 246]$	30 $[14 - 46]$	76.34% ($\pm$5.26%)
Count of XML elements	87 $[24 - 164]$	18 $[7 - 32]$	76.854% ($\pm$10.184%)

Table 5.8: Comparing LFF and ANSI/CEA-2018 models

The count of lines and the count of XML elements demonstrate the reduction in manual coding size and hierarchy depth from the standard models to the LFF design. The count of lines is a measure of the size of the models. The more XML elements there are, the deeper and more complex the model is.

The reduction may also be seen as how powerful the transformation rules are. Going from a model consisting of, on average, 30 elements to a model which has 139, is the result of the thorough exploration of the LFF models by the XSLT rules, looking for maximum interaction possibilities to fulfill the same service requirements.

Augmenting the abilities of deterministic models

Along the course of this thesis, two users sessions allowed the collection of interaction data to train stochastic components. Still, the chosen dialog modeling paradigm is deterministic, based on manual task hierarchies.

Stochastic models such as MDPs and POMDPS are trained from large amount of data collected with real-users recordings. The cost, in terms of time and labor, that accompanies such a collection may explain why these paradigms so far remained in the world of research and were not adopted by commercial companies.

Moreover, stochastic learning algorithms are dependent on the data available and its relevance to the application. In other words, updating a stochastic model or porting it to new domains is, at the moment, an active research area.

Given these considerations, a deterministic way of modeling dialogs seems appropriate. However, unless one wants to implement a very constrained system, deterministic models are difficult to build, debug, maintain and/or modify.

The LFF language offers a light and intuitive syntax, which matches the constraints of the service one wishes to add a spoken interface to.

Building a complex ANSI/CEA-2018 model via the LFF tool is quick and automatic, thanks to the XSLT rules which bridge the gap from one to another.

The LFF tool applies an exploratory algorithm, i.e. it tries to build a task model description that allows for all the dialog paths compliant with the LFF-defined model. In other word, the sequencing order is shuffled, some tasks are combined, etc., so that the user can perform the same dialog in any possible fashion with the same results.

5.7.3 Characteristics summary

Here is a summary of the characteristics of the LFF language.

- No data requirements
- Facilitated design
- Quick conception
- Low expertise requirements
- Guarantee of dialog completion
- Direct domain enlargement
- Automatic dialog variability

5.7.4 Compared to...

In the following table (Table 5.9), the characteristics of the LFF have been mapped to the state of the art dialog modelling methods.

	No data requirements	Facilitated design	Quick conception	Low expertise requirements	Guarantee of dialog completion	Direct domain enlargement	Automatic dialog variability
Flow charts	✓	-	✓	✓	✓	✓	×
Example-based	×	-	×	✓	×	×	✓
Information state	✓	✓	✓	×	×	×	✓
Adjacency pairs	✓	-	✓	✓	×	✓	✓
Stochastic models	×	✓	×	×	✓	×	✓
Linked-form filling	✓	✓	✓	✓	✓	✓	✓

Table 5.9: Characteristics mapping for dialog modelling methods
✓: has the characteristic
×: does not have the characteristic
-: not relevant

5.8 Conclusion

This chapter focused on the DM, which is the decision-making component of an SDS. Several methods for this task were discussed. The current platform integrates a deterministic task-based DM, whose core algorithm is provided by the Disco library. An LFF paradigm was proposed to facilitate the development of models.

The deterministic modeling of dialogs was selected for mainly two reasons. The first one is that stochastic models needs in-domain data to learn a strategy. This data collection is costly. Moreover, modifying a stochastic DM requires to modify the data it has been trained on. Expanding dialogs requires adding more data. Such is often not available. Despite the user-centered approach applied in the here presented work, this two-step process, where one first collects data and then learns the models, is not efficient enough.

However, there is a trade-off in building task hierarchies. The complexity of the system, i.e. its apparent intelligence and flexibility, hinders the easiness of the development. In other words, the larger and more complex the system is, the harder it is to update, maintain, debug, and modify it.

The LFF language offers a light, intuitive syntax to model the application specifications, which are mined by the XSLT rules to produce complex task hierarchies. It tries to combine the best of both worlds.

The LFF reduces the size and the complexity of models and is automatically

transformed. Thus, the conception is facilitated without losing the intelligent aspects of the SDS.

Towards the end of the vAssist project, which provided the application space for the work presented in this thesis, the GUI application, and consequently the dialogs, were reshaped according to the results gained from the previous real-users lab rials. This LFF design proved to be an efficient method to rapidly develop the new models.

5.9 Future work: proposal For the Evaluation of Dialog Management Methods

Evaluating a DM is not a straightforward task. Unlike most classification processes whose performances can be measured by comparing the test outputs with a static ground truth, a manager runs a dynamic algorithm. The corpus collected with a DM may not be suitable for another one. Indeed the response from such a component depends on two parameter sets. One is the inputs it gets, the other one is the maintained inner dialog context. In order to evaluate a system, a pool of live users is necessary.

Also, when comparing two systems, they have to share the same domain(s) for the comparison to be relevant.

In this section, the upcoming vAssist field trials and the work in progress to set up and compare a statistical DM are described.

5.9.1 vAssist Field Trials

In the last months of the vAssist project, the whole system will be deployed to users for an extended period of time. These trials will take place in Austria and in France with elderly users satisfying the target group requirements, i.e. seniors suffering from chronic diseases and persons suffering from (fine) motor skills impairments. Here is an overview of the system.

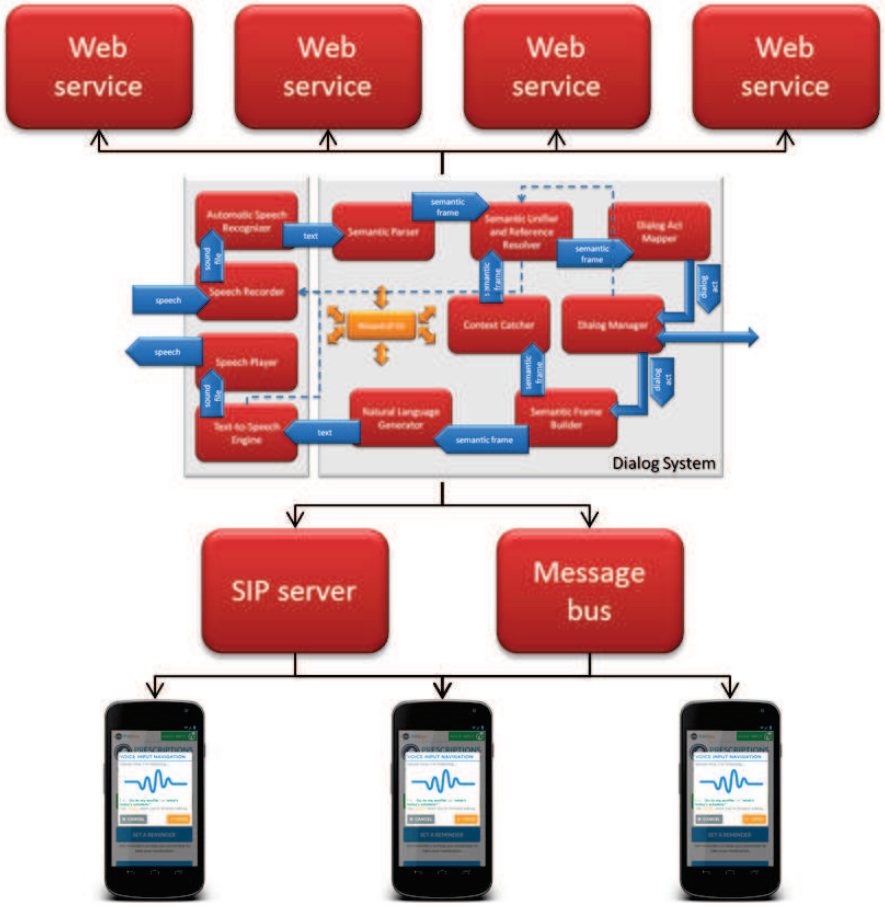


Figure 5.14: vAssist field trials setup

Users access the system via their smartphones, tablets or TVs. They will be regularly prompted and reminded to use it, especially in the first days of the experiment.

Over the course of the experiment, the inter-component data will be automatically recorded, aligned and stored similarly to the previous data collections.

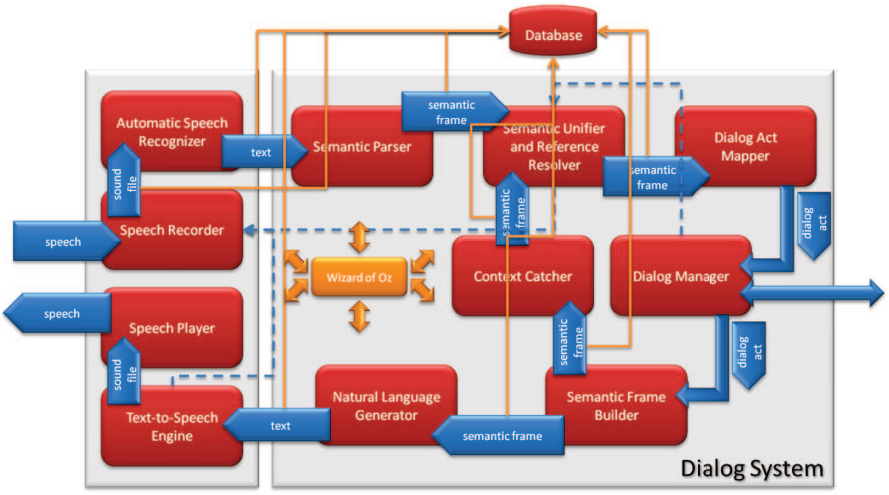


Figure 5.15: Data collection setup

5.9.2 Switching Dialog Managers

The RavenClaw DM is a task-based DM similar to the Disco library. Currently, dialog models are being implemented for the vAssist tasks. Also, RavenClaw's communication protocol, part of the Olympus architecture, is being augmented to send and receive messages to and from an ActiveMQ broker. This aims at integrating the DM into the platform architecture.

During the field trials, the DM will be switched on a regular basis. The objective is to compare the behaviors of both managers for the same task. Metrics such as the average number of turns to achieve tasks and subjective assessment measures (SUS, SASSI, SEQ) will be obtained from the trials.

5.9.3 Comparison With a Statistical Dialog Manager

The Speech Interactive Research group of the Universidad del Pais Vasco/Euskal Herriko Uniberstitatea has been working on the development of a statistical DM [64, 192, 193]. In this paradigm, stochastic dialog models are trained from data. They encode the dialog paths as sequences of system's state nodes and user's input transitions. The graph structure as well as the transition probabilities are automatically learned from a corpus of dialogs.

The collected data (cf. Section 2.7) has been used to train both a user model and the statistical DM. Dialogs were generated from the interaction between these two components. Currently, the new dialog corpus is being analyzed to detect wrong node transitions and improve the learning algorithm

This DM will be integrated into the platform as an alternative DM so that experiments could take place with real users. They will be requested to perform a set of scenarios for each DM method. The DM paradigm of the SDSs they

will be engaging in a conversation with will be transparent to them. Objective and subjective evaluations will be performed and analyzed.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This document reports on 3 years of research work conducted at Télécom ParisTech whose main objective was to build an SDS for a particular group of users, i.e. elderly people from France, Austria and Italy.

The path of research followed the requirements of the supporting projects CompanionAble, Arhome and vAssit. These projects gave access to users at the specification, development and deployment phases, and hence helped to design the system, collect data, get feedback and measure performances in real conditions. This user-centered process brought benefits to this work and anchored it into the reality of SDSs.

In this endeavor, multiple areas and domains have been explored, creating an overall idea of the SDS design and, while focusing on some components, keeping in mind the interaction between them, thus not restricting the scope to a single functionality.

Collaborating with experts from different domains highlighted the need for tools to facilitate the configuration of SDSs which are complex multi-task processing systems.

While the human-machine interaction aims at freeing the communication from a constraining sequence, experiments showed that some users, especially elderly, while keen to use the system, expected it to be highly reliable and robust. That request appeared as early as the first WoZ-based experiments.

In order to address the three main research questions listed below and in chapter 1, this document is divided into four parts or application domain to which this thesis brings a contribution. The problematics are:

- How can we make the human-machine spoken interaction more reliable?

- What are the important features for an SDS to improve its human-like appearance?
- How can we support the development of SDSs?

Following are the main contributions of this thesis.

A modular open-source platform for spoken dialog systems The platform detailed in chapter 2 has three usages.

The owner of a service who wants to add a vocal interface to it may set up the platform so that it handles the dialogs to access the service. Individual components are easy to set up and this task does not require in-depth knowledge of the algorithm they implement.

Researchers from the SDS-related domains such as ASR, NLU, DM, NLG and TTS often specialize in one specific functionality and thus lack the overall system to integrate their technology in. Since the platform is modular, any component (or set of) can be replaced by another one whose inputs and outputs are identically formatted. Thus, the in-context performance of a new algorithm can be evaluated within the framework.

Finally, the most obvious way of using the platform is by interacting with it. It has been ported, with a set of configuration files, to a server accessible via a VoIP link, i.e. any SIP client with an authorized ID can call the system to experiment with it.

A Step Towards continuous listening for spoken interaction Elderly users, as it was observed in early experiments with the system, tended to forget about triggering the recording of segments, i.e. they addressed the system like it was a human being. Continuous listening solves that but brings up many more issues with respect to SDS design and development. An optimal trade-off had to be found between how constrained the system is and how much it allows the user to control the interaction.

A first iteration of a continuous listening method showed encouraging results, although it needed to be set up for a specific environment and a specific speaker.

Currently, the platform implements a lighter mechanism that is shared among dialogs and users and spans over more components of the ASR and NLU subsystems. This continuous listening is not yet as fine-grained as one would wish but it is a step towards a more natural manner of talking to a machine.

A sub-system to map natural-language utterances to situated parametrized dialog acts The SDS framework developed here includes a modular NLU system, which segments the incoming signal, transcribes the speech, parses the utterances, augments the inputs with the dialog context and additional external sources, selects the user DA, and tries to detect and recover from errors.

The NLU components have been set up for three languages using the support of native speakers. Since many languages were to be implemented, the subsystem has been conceived to be as portable as possible. Thus, only the front-

end processors are language-dependent while the remaining ones are based on semantic concepts, whose human-readable representations are English words.

The four mandatory features of an NLU system to be called as such have been integrated:

1. allow variations in the utterances of the user
2. allow for a mixed initiative
3. integrate the dialog context in the process
4. integrate relevant external environment variables to augment the information

The linked-form filling language: a new paradigm to create and update task-based dialog models The LFF modeling paradigm enables non-expert developers to build and maintain dialog task hierarchies compliant to the ANSI/CEA-2018 standard, while hiding the complexity of the final models. The tool adds to the domain of DM design by opening it up to a wider community. It is an alternative to the data-based learned models which require less development efforts but much more data collection/annotation phases and are, at the moment, not mature enough to be put to practical use.

All the dialog models currently part of the platform have been transformed from an LFF file, itself encoding the services' specifications

6.2 Future Work

The framework is a great baseline platform to conduct experiments with SDSs. The current setup allows to do so in three different languages: German, French and Italian. Here are ideas for the future to build upon that.

There are currently a dozen services that the server-based SDS provides an interface for. I wish to offer more of them to users.

Also, implementing new languages would demonstrate the above claims. Currently, a Spanish version is in progress. The designer is creating the NLG templates file and an SP corpus for this language.

The current components of the platform are not the initial ones for most of them. Actually, the system is still evolving. New components should be experimented such as an improved ASR module, additional NLU resources, stochastic DMs, a trained NLG, etc. The statistical DM is an instance of such alternative components.

One drawback of the system is that it lacks, in some dialogs situation, guidances to give to the user. The latter is left without much information about the state of the system or the options available. This should be improved. The SDS is set up for a well-defined domain and some specific services. Out-of-scope utterances are rejected with a flat "I don't understand that" system's turn when the DM has no context information. This limits the naturalness of

the system. Some generic deflection dialogs should be implemented as well as a bit of "chit-chat" handled by the SDS.

This SDS platform may be a data collector to seamlessly shift to a fully stochastic system. For that, it needs data and practical algorithms to learn and execute models. So far, the SDS as been deployed to a specific chunk of the population in Europe. The deployment of the Let's Go Bus Information System showed how rewarding it is to release a system to the general population. For that the automatic dialogs need to answer actual needs and provide services at least as efficiently as their human-based counterpart. Large deployment means more data collected, the ability to learn online and a continuous feedback on the usage of such system. Such an SDS could be available as an interface to the online room booking system of the school, a guide of the buildings for visitors or an interface with the information systems service desk, etc.

Even though machine learning methods have been scarcely used to set up components in the work above, there is no denying it is the future of the interactive computing. I wish to work on the application of such techniques for the ASR, the NLU and the dialog management.

There are two ways to integrate data in an SDS. The offline training uses a corpus of data to train or adapt models. Learning new dialog strategies, utterance structures and/or adapting to the voice of the users online has the same objective but require less manual work, though it seems that it is less reliable.

Another promising area in the human-machine interaction is the adaptation and learning from a specific user. It means either adapting the existing system so that it fit better the user or giving the ability for a user to "teach" the system, i.e. create new tasks, services and increasing the natural language domains via the interaction.

Chapter 7

Publications

1. Pierrick Milhorat, Dan Istrate, Jérôme Boudy, and Gérard Chollet. Hands-free Speech-sound Interactions At Home. In *European Signal Processing Conference*, pages 1678–1682, 2012

2. Pierrick Milhorat, Dan Istrate, Jérôme Boudy, and Gérard Chollet. Interactions Sonores Et Vocales Dans l'Habitat. In *Interactions Langagières pour personnes Agées Dans les habitats Intelligents*, pages 17–30, 2012

3. Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, and Jérôme Boudy. Un Systeme De Dialogue Vocal Pour Les Seniors: Etudes Et Spécifications. In *Journées d'Etude sur la TéléSanté*, 2013

4. Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, and Jérôme Boudy. What If Everyone Could Do It? A Framework For Easier Spoken Dialog System Design. In *Engineering Interactive Computing Systems*, pages 217–222, 2013

5. Stephan Schlögl, Gérard Chollet, Pierrick Milhorat, Jirasri Deslis, Jacques Feldmar, Jérôme Boudy, Markus Garschall, and Manfred Tscheligi. Using Wizard of Oz To Collect Interaction Data For Voice Controlled Home Care And Communication Services. In *Signal Processing, Pattern Recognition and Applications*, pages 12–14, 2013

6. Stephan Schlögl, Pierrick Milhorat, and Gérard Chollet. Designing , Building and Evaluating Voice User Interfaces For The Home. In *Conference on Human Factors in Computing Systems*, 2013

7. Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, and Jérôme Boudy. Multi-step Natural Language Understanding. In *SIGdial Meeting on Discourse and Dialogue*, pages 157–159, 2013

8. Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, Jérôme Boudy, Anna Esposito, and Gianni Pelosi. Building the Next generation of Personal

Digital Assistants. In *ATSIP International Conference on Advanced Technologies for Signal & Image Processing*, pages 458–463, 2014

9. Mathieu Radenen, Pierrick Milhorat, Thierry Artières, Jérôme Boudy, and Gérard Chollet. Etude des HMM paramétriques pour la reconnaissance de parole en environnement bruité. In *Journées d'Etudes sur la Parole*, 2014

10. Hugues Sansen, Jean-Louis Baldinger, Jérôme Boudy, Gérard Chollet, Pierrick Milhorat, and Stephan Schlögl. vAssist : Building The Personal Assistant For Dependent People - Helping Dependent People to Cope With Technology Through Speech Interaction. In *HEALTHINF International Conference on Health Informatics*, 2014

11. Stephan Schlögl, Pierrick Milhorat, Gérard Chollet, and Jérôme Boudy. Designing Language Technology Applications: A Wizard of Oz Driven Prototyping Framework. In *EACL Conference of the European Chapter of the Association for Computer Linguistics*, pages 85–88, 2014

12. Pierrick Milhorat, Gérard Chollet, and Jérôme Boudy. Un Système de Dialogue Vocal Comme Agent d'Aide à la Personne. In *Journées d'Etude sur la TéléSanté*, 2014

13. Gennaro Cordasco, Marilena Esposito, Francesco Masucci, Maria Teresa Riviello, Anna Esposito, Gérard Chollet, Stephan Schlögl, Pierrick Milhorat, and Gianni Pelosi. Assessing Voice User Interfaces: The vAssist System Prototype. In *Cognitive Infocommunications*, pages 91–96, 2014

Chapter 8

Résumé long

Table des matières

8.1 Résumé

L'interaction vocale avec des systèmes automatiques connaît, depuis quelques années, un accroissement dans l'intérêt que lui porte tant le grand public que la communauté de la recherche. Cette tendance s'est renforcée avec le déploiement des assistants vocaux personnels sur les terminaux portables.

Cette thèse s'inscrit dans ce cadre pour aborder le sujet depuis deux points de vue complémentaires. D'une part, celui apparent de la fiabilité, de l'efficacité et de l'utilisabilité de ces interfaces. D'autre part, les aspects de conception et d'implémentation sont étudiés pour apporter des outils de développement aux concepteurs plus ou moins initiés de tels systèmes.

A partir des outils et des évolutions (très récentes) dans le domaine, une plate-forme modulaire de dialogue vocal a été agrégée. Progressivement, celle-ci a été configurée pour répondre aux exigences des scénarios d'usage et de démonstration dans l'optique des collaborations encadrant ce travail. Le système s'est complexifié et est constamment en évolution suivant les deux approches mentionnées plus haut.

L'interaction continue, basée sur une "écoute" permanente du système pose des problèmes de segmentation, de débruitage, de capture de son, de sélection des segments adressés au système, etc... Une méthode simple, basée sur la comparaison des résultats de traitements parallèles a prouvé son efficacité, tout comme ses limites pour une interaction continue avec l'utilisateur.

Les modules de compréhension du langage forment un sous-système interconnecté au sein de la plate-forme. Ils sont les adaptations d'algorithmes de l'état de l'art comme des idées originales. Ils ont été pensé pour rendre l'interaction naturelle et fiable tout en limitant la complexité de leur configuration et en maintenant leur généricité et donc leur usage à travers plusieurs dialogues. L'utilisabilité est évaluée à partir de données collectées lors d'essais en laboratoire avec des utilisateurs réels. L'aisance dans la configuration d'un tel système et sa modularité, plus difficiles à prouver empiriquement, sont discutées.

Le choix de la gestion du dialogue basé sur des modèles de tâches hiérarchiques, comme c'est la cas pour la plate-forme, est argumenté. Ce formalisme est basé sur une construction humaine et présente, de fait, des obstacles pour concevoir, implémenter, maintenir et faire évoluer les modèles. Pour parer à ceux-ci, un nouveau formalisme est proposé qui se transforme en hiérarchie de tâches grâce aux outils associés. Ce document se veut être une référence du nouveau langage code et de sa conversion, il présente également des mesures d'évaluation de l'apport d'un tel outil.

Construire un système de dialogue vocal peut se faire selon deux méthodes: soit on cherche à créer un système utilisable par le plus grand nombre, soit on cible un groupe d'utilisateurs particulier. En terme de fiabilité, un système générique large perd en précision dans ses traitements puisque configuré/appris sur une moyenne de ses utilisateurs potentiels. En revanche, les utilisateurs

obtiennent un accès immédiat aux services proposés. A l'opposé, une interface adaptée et/ou adaptive demande à l'utilisateur de s'investir dans la configuration et l'apprentissage du système pour en retirer des bénéfices postérieurs. Dans cette thèse, cette dernière approche est favorisée: des mécanismes de personnalisation permettent d'adapter un système général à un utilisateur spécifique. Plusieurs expériences montrent l'amélioration des performances du système de dialogue vocal dans le cadre d'un usage personnel.

8.2 Introduction

Récemment, de nombreuses firmes internationales de technologie ont déployé leur propre assistant virtuel personnel. Ces produits ont attiré l'attention du grand public de par leur interaction naturelle (vocale) mais ont, d'autre part, essuyé des critiques concernant la fiabilité, l'utilité réelle, la protection des données personnelles ou encore les aspects propriétaires de ces interfaces.

De son côté également, la communauté scientifique, bien que saluant l'aboutissement démontré de la technologie, déplore le manque d'ouverture des systèmes. En effet, les chercheurs bénéficieraient de l'accès aux outils développés, tireraient des leçons de l'étude de ceux-ci et pourraient comparer leurs résultats avec ces assistants à l'échelle du composant si de tels technologies étaient ouvertes.

C'est dans ce contexte que s'inscrit ce travail de thèse. Le principal résultat en est l'élaboration d'une plate-forme de dialogue vocal ouverte qui apporte des éléments de réponse aux problématiques suivantes:

- Comment fiabiliser l'interaction avec les systèmes de dialogue vocal?
- Quelles caractéristiques sont importantes pour rendre une apparence naturelle et spontanée à l'interaction vocale homme-machine?
- Comment soutenir le développement des systèmes de dialogue vocal par des concepteurs plus ou moins initiés?

La plate-forme de dialogue vocal Un système de dialogue vocal se définit par une interface vocale (en langage naturel) qui permet l'accès à un ensemble de services [35]. Une telle interface allie les technologies de capture et d'analyse du signal [5, 8, 194, 195], de reconnaissance des formes [90, 152], d'apprentissage automatique, d'intelligence artificielle [29, 37], de traitement du langage [7, 48], de génération du langage, de synthèse vocale [141, 177], etc...

On parle de dialogue dès lors que l'interaction excède un tour, i.e. dès lors que le système et le ou les utilisateur(s) construisent l'interaction sur une base d'information partagée. De fait, un système de dialogue requiert un gestionnaire [88] qui sélectionne, à chaque tour, la ou les actions les plus appropriées en fonction de l'historique du dialogue et de sa modélisation.

Plusieurs équipes de recherche ont proposé des plate-formes de dialogue vocal, partielles ou complètes, ouvertes ou fermées, configurables ou statiques

[61, 73, 98, 143, 188]. L'une des plus aboutie étant, de mon point de vue, Olympus [17]. Celle-ci définit un protocole de communication et associe plusieurs modules pour construire un système de dialogue vocal. Olympus fut utilisé pour créer RoomLine, MeetingLine, TeamTalk [74], ConQuest [16], Let's Go! Bus Information System [57, 157, 158], etc... Le dernier d'entre eux, Let's Go, est accessible depuis 2003 par l'intermédiaire d'une liaison téléphonique. Il connecte les usagers à un service automatique d'information sur les lignes de bus de Pittsburgh et de ses environs.

Préalablement, le MIT avait proposé JUPITER [223] tandis que Philips construisait le automatic train timetable information system [6]. Let's Go, basé sur l'architecture Olympus a été pensé pour être ouvert: ses sources comme les données d'usage collectées sont disponibles. Cependant, tous les modules ne sont pas transférables à de nouveaux scénarios, d'autres ne sont pas adaptables et doivent être reconstruits et, plus encore, remplacer un composant du système s'avère difficile du fait de l'éparpillement de la documentation et/ou de son absence.

Le document joint renseigne la plate-forme construit en fil rouge de cette thèse. Celle-ci se veut complètement ouverte et aisément configurable pour des dialogues orientés vers l'accès à des services (par opposition aux agents conversationnels).

Écoute permanente et robustesse de la reconnaissance de la parole L'interaction vocale avec une machine peut être caractérisée par le degré de contrôle de l'écoute. Le terme englobe l'ensemble des paramètres relatifs à l'enregistrement du signal. Dans le cas où elle est très contrainte, l'utilisateur doit signaler sa volonté de s'adresser au système et se plier aux exigences de la capture du son. Pour ce faire, la majorité des systèmes, se servent d'une modalité binaire (bouton, geste, ...) ou cherche en permanence une phrase clé dans le signal enregistré (on peut alors parler d'écoute semi-permanente). Les performances d'un système et son efficacité dépendent alors essentiellement de la qualité de l'enregistrement: distance de l'utilisateur au microphone [121], souffle direct sur le capteur, bruit ambiant, écho, utilisation d'algorithmes de pré-traitement, quantité de capteurs, etc...

Contraindre les conditions d'enregistrement permet d'améliorer la robustesse du système mais en ternit les aspects naturels. De fait, l'écoute permanente et libre présente des avantages de ce point de vue, en dépit d'une fragilité accrue du système [7].

Nous verrons dans la section 8.4 comment une méthode basée sur une reconnaissance automatique de la parole parallèle et une comparaison des résultats permet la validation ou le rejet de segments de signal dans le cadre d'une écoute permanente.

Compréhension du langage appliqué au dialogue vocal Pour obtenir une interface en langage naturel, deux sous-systèmes, l'un interprétant le langage de l'utilisateur (compréhension) et l'autre générant les tours de parole du

système (génération), doivent être apposés de part et d'autre du module pivot qu'est le gestionnaire de dialogue. Nous nous intéressons à la compréhension, ici, dans le cadre des systèmes de dialogue vocal. Celle-ci a pour but d'extraire un sens, une signification pour le système de dialogue, dans une entrée parlée.

Les premières interfaces de dialogue interprétaient les segments de texte issus de la reconnaissance automatique de la parole par la détection de mot clés (ou séquence de) [48, 204]. Vinrent ensuite les grammaires déterministes suivies des grammaires stochastiques pour lesquelles un important travail préalable d'analyse et de conception est nécessaire [53, 68, 132, 201, 202, 203]. L'avènement des méthodes d'apprentissage automatique dans plusieurs domaines du traitement du signal ou de l'intelligence artificielle a débordé sur cette tâche et des systèmes basés sur les Modèles de Markov Cachés (MMCs), tels que Chronus [12, 144, 145], ont vu le jour. Ces modèles sont appris à partir de données d'interaction, avec les avantages et les inconvénients inhérents à ces méthodes. Un modèle intéressant, le Hidden Vector State [76, 77, 78, 180], s'appuie sur les MMC pour proposer un algorithme de parsing moins gourmand en annotation.

Il est nécessaire de prendre en considération des phénomènes inhérents au langage parlé, tels que les hésitations, les répétitions, les erreurs de grammaire, lors de l'analyse sémantique de segment de parole transcrits [13, 14, 51, 187]. Ainsi, il est préférable de procéder à un parsing partiel qui s'attache a étiqueter une partie de la transcription au lieu d'un parsing total pour lequel à tous les mots doit correspondre un symbole sémantique. C'est le choix qui est fait pour le sous-système de compréhension du langage de la présente plate-forme.

Un autre aspect de la compréhension du langage dans le contexte du dialogue est la notion de dynamisme dans l'association du sens sémantique d'un segment de parole à l'acte de dialogue. En effet, cette correspondance dépend de l'état du dialogue et de son historique. La plate-forme développée intègre des mécanismes de résolution de références aux concepts du dialogue précédemment abordés et de références à un savoir partagé tel qu'on pourrait l'envisager lors d'une interaction entre deux entités humaines.

Modélisation des dialogues: linked-form filling Le chapitre final de ce manuscrit présente les efforts consentis pour faciliter la modélisation des dialogues. Les modèles de dialogue, ceux sur lesquels le gestionnaire base son raisonnement, ont suivi une évolution similaire à celle du domaine de la compréhension du langage.

Partis de scripts de réponses prédéfinies [204], la tendance actuelle en matière de gestion du dialogue est dominé par les POMDPs [25, 28, 79, 94, 167, 191, 207, 218, 221] qui présentent une base stochastique intégrant l'incertitude des résultats du traitement du langage dans la modélisation.

Entre les deux, l'Information State (IS) [101, 134] a été proposé. Il définit l'état du dialogue comme la somme des informations échangées au cours de celui-ci. Un ensemble de règles, conditionnées par leur applicabilité et une stratégie de mise à jour, assurent la gestion du dialogue.

La modélisation de l'état du dialogue par cumul des variables utiles à celui-ci

a inspiré Lison [115, 116, 117] qui étend les règles de mise à jour en y injectant des aspects stochastiques.

Les modèles basés sur les hiérarchies de tâches [18, 71, 112, 161, 162, 163, 164, 165, 182, 185] obéissent au principe suivant: une tâche, si elle n'est pas primitive, est complétée après que l'ensemble de ses tâches filles l'ai été, une tâche primitive s'exécute indépendamment. Définir les tâches et leur hiérarchie fait appel aux capacités d'analyse des concepteurs et l'anticipation de ceux-ci quant au déroulement du dialogue.

Ce paradigme, contrairement au MDPs et POMDPs, ne s'appuie pas sur une collection de données préalable pour établir les paramètres. Cependant, il participe grandement à la complexification des SDVs, à la difficulté de leur maintenance et de leur portabilité pour les concepteurs du système, plus encore pour les non initiés.

Pour autant, les modèles de tâches permettent la création de modèles de dialogue sans collection de données préalable ou bien dans le but de collecter ces données auprès d'un groupe initial d'utilisateurs potentiels.

Cette thèse propose de supplanter les modèles bruts avec une nouvelle abstraction appelé Linked-Form Filling (LFF) qui modélise un service selon un ensemble de formulaires inter-connectés. Ceux-ci sont ensuite transformés en hiérarchies de tâches à l'aide de l'outil de transformation approprié.

La suite de ce mémoire comporte quatre parties. Dans la première, la plate-forme open-source de dialogue vocal est introduite: son architecture et le fonctionnement de chacun des modules la composant. Ensuite, viennent en détails les aspects de robustesse de la transcription vocale et de gestion de l'écoute du système. Suivent ensuite les travaux sur la compréhension du langage et le nouveau paradigme LFF de modélisation du dialogue par formulaires inter-connectés.

8.3 La plate-forme de dialogue vocal

8.3.1 Introduction

Les plate-formes disponibles aujourd'hui pour le développement de SDVs sont peu nombreuses et les systèmes commerciaux, jusqu'à présent, n'offrent que peu voire aucune option de configuration [6, 17, 36, 68, 98, 158, 188, 215, 223].

L'usage de la reconnaissance vocale, pourtant, a pu être généralisée par la mise à disposition d'APIs [173] et de moteurs de reconnaissance, ouvrant cette modalité aux services développés sur les machines portatives mais également les explorateurs internet intégrant un dispositif de capture de son [7, 133, 183]. Ces services se basent sur d'immenses quantités de données et se nourrissent de celles collectées lorsqu'un utilisateur y fait appel, autrement dit, le cercle vertueux de l'apprentissage automatique (figure 8.1) a été initié.

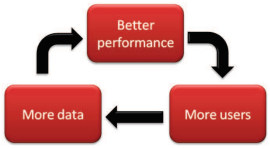


Figure 8.1: Cercle vertueux de l'apprentissage automatique

L'avènement de ces interfaces a élargi le champ des possibilités dans le domaine de l'interaction vocale avec une machine et les intérêts se recentrent sur la question: "Comment bien utiliser ces transcriptions du langage naturel?" [67].

Un système pour l'interaction homme-machine tend à reproduire les comportements humains, que ce soit pour animer un robot humanoïde ou pour engager une discussion avec un avatar virtuel. Le dialogue vocal avec une machine demande de comprendre le sens des mots au-delà de leur définition immédiate. Il faut prendre en compte leur combinaison, les figures de styles, les sous-entendus et le contexte dans lequel un énoncé est exprimé. Les SDVs ajoute cette couche d'analyse à la reconnaissance automatique de la parole pour interagir avec un utilisateur dans le but de l'accompagner dans l'accomplissement d'une tâche.

Ce chapitre présente la plate-forme développée tout au long de cette thèse. Après une vue d'ensemble de celle-ci, dans son état actuel, un exemple permettra de détailler les fonctions de chacun des modules la composant.

8.3.2 Vue d'ensemble

Au cours du travail présenté dans ce document, une plate-forme intégrant des modules utiles à un SDV a été assemblée avec pour but premier de constituer une base modulaire pour l'étude de tels systèmes et de leurs composants mais aussi dans le but d'expérimenter ce type d'interaction et ses applications.

Pour ce faire, le protocole de communication entre les modules est léger et décentralisé, i.e. il n'existe pas de régulateur ou de noeud central de gestion des messages. Tous les composants agissent comme des services indépendants asynchrones. ActiveMQ est un protocole de transmission de messages sur le modèle client-serveur. Un client producteur émet des messages dont l'entête contient l'identifiant de la destination. Chaque client enregistré comme consommateur pour une destination se voit recevoir les envois vers celle-ci. Ce système flexible et léger permet le remplacement facile de n'importe quel composant ou l'ajout rapide d'un nouveau module.

Le système a connu plusieurs évolutions et, au moment où ce document est rédigé, s'agence comme sur le figure 8.2.

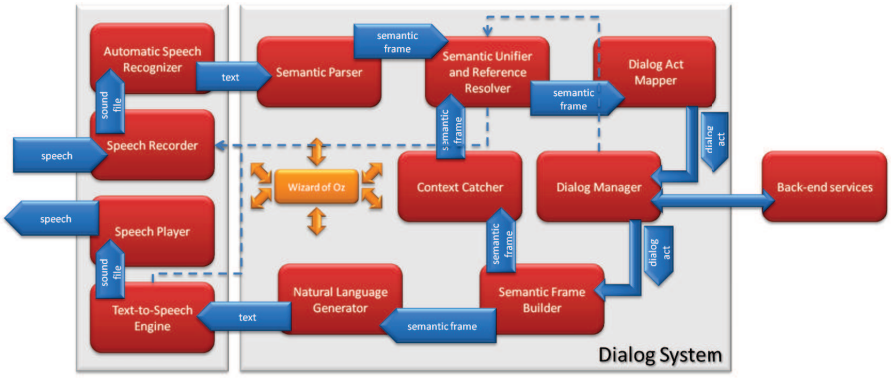


Figure 8.2: Architecture de la plate-forme

8.3.3 Fonctionnement des composants

Basons nous sur un exemple pour détailler les fonctionnalités de chaque composant et suivre la séquence de traitement pour compléter un tour d'interaction. Le service simulé permet à un usager de consulter la météo à la date et pour le lieu de son choix (en considérant que cette information est disponible pour tous les lieux et toutes les dates). Le niveau de détail des données retournées est réglable selon deux niveaux: soit haut (température, force du vent, hygrométrie, pression), soit bas (météo générale).

Voici les étapes de traitement:

1. L'utilisateur dit: "Quel temps fait-il aujourd'hui?". Le système écoute en permanence, l'utilisateur n'a pas besoin de signaler son intention de s'adresser au système.

2. Le module d'enregistrement segmente le signal continu reçu selon un seuil de niveau sonore (en dB), un délai de silence (une fenêtre du signal dans laquelle l'énergie est inférieure au seuil) et un durée minimume. Le résultat est un fichier sonore contenant un tour de parole de l'utilisateur. Une fois ce dernier obtenu, le composant se met en attente de messages de réactivation.

3. La reconnaissance de la parole produit une liste classée de, au plus, 7 hypothèses sur la transcription du segment de parole. Toutes les hypothèses sont transmises au composant suivant du SDV: l'analyseur sémantique.

4. L'analyseur sémantique associe une trame sémantique, i.e. un représentation de la signification, à chaque transcription supposée. Cette analyse s'appuie uniquement sur la définition et la combinaison des mots dans la phrase.

5. L'unificateur sémantique et résolveur de référence a plusieurs rôles. Il peut:

- rejeter le tour de parole de l'utilisateur si le score de confidence de la meilleure hypothèse de transcription est trop bas: soit en l'ignorant "silencieusement", ce qui a pour effet d'abandonner le traitement et de réactiver l'enregistreur, soit en déclenchant un dialogue spécial qui signale à l'utilisateur l'incompréhension et que ce dernier devrait reformuler sa requête.
- résoudre les références relatives (comme "aujourd'hui" dans le cas présent) en leur associant des valeurs absolues.
- résoudre les références relatives au dialogue. En effet, l'analyse sémantique se basant uniquement sur les mots, elle n'inclue pas les notions de contexte du dialogue. La section 8.5 détaillera cette fonctionnalité.
- réajuster le niveau ou l'espace sémantique de l'analyse

Le SURR ne transmet, dans un premier temps, que le résultat du traitement de la meilleure hypothèse de reconnaissance de la parole. Les autres résultats sont conservés.

6. Une correspondance entre la trame sémantique et un acte de dialogue parmi ceux extraits du gestionnaire de dialogue est obtenue au sein du dernier composant préalable au gestionnaire. Celle-ci peut ne pas exister, i.e. l'entrée de l'utilisateur, telle que transcrite par le module de reconnaissance automatique de la parole n'a pas d'effet sur le dialogue. Dans ce cas, une requête est envoyée au SURR qui fournit la trame sémantique correspondant à l'hypothèse suivante dans la liste classée originale. Si cette liste est épuisée, la meilleure trame est à nouveau transmise avec un drapeau "final".

7. Le gestionnaire combine l'historique du dialogue, le ou les modèles disponibles et le dernier acte de dialogue de l'utilisateur pour sélectionner une action du système. Il produit en sortie un acte de dialogue système.

8. L'acte de dialogue système est converti en trame sémantique.

9. A partir d'une trame sémantique, le générateur de langage naturel instancie un modèle paramétrique qui exprime le sens porté par la trame de manière à ce qu'un humain puisse le comprendre.

10. La synthèse de la parole génère de la voix à partir du texte. dans cet exemple, le tour système est: "Pour quelle localité souhaitez-vous connaître les conditions météorologiques?"

11. Le fichier de parole synthétisée est joué pour l'utilisateur, cela clos un tour.

Voici un exemple du déroulement d'un dialogue pour ce service

1. Utilisateur: "Quel temps fait-il aujourd'hui?"

2. Système: “Pour quelle localité souhaitez-vous connaître les conditions météorologiques?”

3. Utilisateur: “Pour paris”

4. Système: “Quel niveau de détail souhaitez-vous?”

5. Utilisateur: “Peu de détails”

6. Système: “La météo est ensoleillée, il fait 25°C”

Un SDV est une interface avec un service via une interaction intelligente. Pour implémenter l'interaction, il faut connecter les deux entités: le service et le SDV. La plate-forme intègre, au niveau du gestionnaire de dialogue, des espaces de code JavaScript qui permettent de faire appel à des APIs, des services web, etc... Dans nos scénarios, une connection à un bus de données, à des services web et à une application graphique est maintenue à l'aide de tels scripts.

8.3.4 Configuration

Dans la dernière partie de cette section, la configuration du système et la méthode avec laquelle nous avons procédé est détaillée. Le séquence de traitement naturelle du dialogue est inversée pour ces explications.

MaryTTS [141, 177] est une plate-forme de synthèse vocale maintenue par le Cluster of Excellence MMCI et DFKI. L'équipe de développement du système propose plusieurs modèles selon la langue et la voix que l'on souhaite générer, ainsi que des outils de construction/manipulation de modèles. Il faut, pour utiliser le moteur, créer un serveur sur lequel on peut alors placer des appels contenant le texte à synthétiser et d'autre informations prosodiques et lire le flux de données retourné. Pour son intégration, un client MaryTTS basique a été augmenté, intégrant un client ActiveMQ et un lecteur de fichiers sonores.

La génération du langage naturel est basée sur des modèles qui se différencient par le but, le nom et la valeur des slots. Il peut exister plusieurs modèles pour une même trame sémantique, la sélection est alors faite aléatoirement entre ceux-ci.

Le gestionnaire du dialogue et le sous-système de compréhension du langage ne sont pas détaillés ici puisqu'ils le seront dans les chapitres suivants.

Bien que l'implémentation actuelle du système utilise une API distante (à savoir Google Speech API [173]) pour la tâche de reconnaissance automatique de la parole, les premières versions de la plate-forme intégraient Julius [104, 105], un moteur de reconnaissance développé au laboratoire Kawahara de Kyoto et configuré avec des modèles appris sur des corpus issus de la presse écrite et radiophonique francophones. Le module implémente la méthode décrite en section 8.4 pour différencier, fiabiliser voire corriger les transcriptions.

Le passage du module initial qui remplissait les fonctions d'enregistrement et de reconnaissance à deux composants, reliés par l'intermédiaire du protocole ActiveMQ, a accru la flexibilité du système.

8.4 Écoute permanente et robustesse de la reconnaissance de la parole

8.4.1 Introduction

Le domaine de la reconnaissance de la parole a connu de nombreuses évolutions jusqu'à présent, que ce soit pour la paramétrisation du signal, la sélection des symboles, la modélisation des unités acoustiques ou encore l'agrégation des unités modélisées. Certaines étapes ont eu plus d'impact que d'autres telles que la proposition des MFCCs [46], l'introduction des HMMs pour la modélisation [59, 90, 151, 152, 154, 211, 219] et l'implémentation des n-grammes [33, 34, 139, 181, 186, 222] pour remplacer les grammaires déterministes. Malgré ces progrès, la tâche reste encore très dépendante des conditions d'enregistrement, de la qualité et de la quantité des données d'apprentissage. Depuis peu, le secteur est dominé par des entreprises mondiales qui traitent et collectent d'immenses masses de données.

Il existe un compromis lors de la construction d'un système de reconnaissance ou de classification: augmenter le nombre de classes ou l'espace de recherche influe sur la précision et le rappel de manière inversement proportionnelle. En d'autres termes, un large système sera moins précis pour une tâche donnée mais offrira plus de nuances/variations tandis qu'un système restreint conservera une précision importante sur les données d'apprentissage mais se dégradera vite lorsqu'il est nécessaire de généraliser ou d'étendre à d'autres domaines. C'est de ce principe, qui suggère que l'équilibre d'un système se situe entre les deux extrêmes, que la méthode proposée ici s'inspire.

Dans un premier temps, les conditions de la tâche sont présentées, avec une analyse des conséquences sur un système de reconnaissance vocale. Sera ensuite décrit la méthode proposée, la théorie de son application pour finir par une évaluation chiffrée de ses capacités.

8.4.2 Problématiques

Le module de reconnaissance de la parole avait pour objectif d'être intégré à un système réparti dont le noeud principal est un robot assistant autonome et mobile au sein d'une maison connectée. La prise de son est effectuée avec un microphone CMT (Coincidence Microphone Technology) omnidirectionnel et sans fil. Celui-ci est placé au sommet du robot compagnon, soit à environ 1,5m du sol. Le son est capté et transmis en continu vers le module de traitement audio dont la tâche est de segmenter le signal, faire une hypothèse unique sur le contenu vocal des segments et transmettre celle-ci au gestionnaire de dialogue multi-modal.

Cette configuration a plusieurs conséquences sur la manière de traiter le flux continu de parole et de bruit capturé par le microphone.

La première d'entre elle est la distance au locuteur qui peut varier de quelques centimètres à une dizaine de mètres (selon la configuration et la taille de l'habitation

concernée).

La réverbération sur les murs, les plafonds, les sols et les objets de taille conséquente ajoute des signaux parasites convolués et retardés.

A ces problématiques spatiales vient s'ajouter celle des bruits de l'environnement: les médias (radio, télévisions), les appareils ménagers (machine à laver, four à micro-ondes, aspirateur), les bruits de portes, de clés, d'eau, de pas, de toux, les bris de verre, la pluie sur les fenêtres et le toit, etc... Il existe une solution de filtrage dans laquelle on modélise les bruits nuisibles susceptibles d'être entendus et on tente de les reconnaître dans le flux et de les éliminer [199]. D'autres équipes de recherche ont placé des microphones aux abords des sources sonores majeures pour en soustraire l'influence dans le signal perçu [102, 103].

Enfin, le son provenant des moteurs du robot sur lequel est placé le microphone et de ses haut-parleurs peut être directement annulé du fait du contrôle sur leur activité.

Contrairement à d'autres prototypes abordant les mêmes problématiques, le système ne possède que d'un unique capteur mobile et de fait ne permet pas les techniques de séparation de sources basé sur les réseaux de microphones ou la modélisation de l'espace.

Pour finir, la dernière question qui se pose concerne l'attention du robot. En effet, en admettant que le système filtre parfaitement le bruit environnant, les effets d'écho et ramène l'amplitude à une distance constante, ce signal propre de parole pourrait correspondre à une discussion entre deux personnes présentes dans la zone de capture du microphone, à un appel téléphonique, à un débat télévisé ou tout autre énoncé non destiné à une interaction avec la machine. Il est donc indispensable de créer des mécanismes qui contrôle l'attention du robot et ce de manière transparente et vocale.

8.4.3 Méthode d'écoute continue

La méthode proposée est construite comme suit:

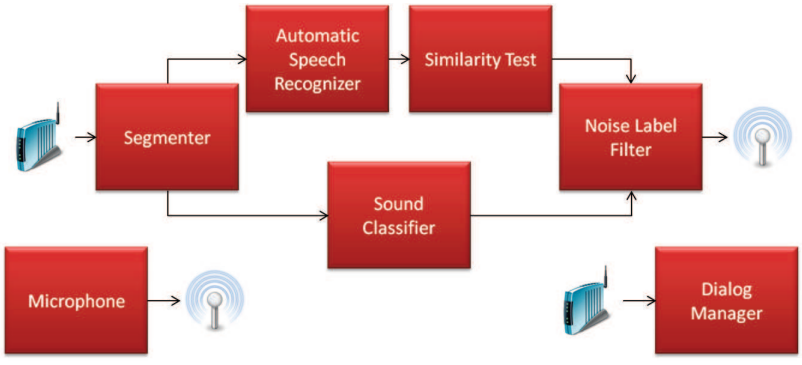


Figure 8.3: Schéma

Le flux audio est segmenté selon un seuil de niveau sonore et un délai de silence.

Un segment se dédouble pour qu'une copie, identifiée par horodatage, suive la branche de reconnaissance de la parole tandis que l'autre, identifiée de la même façon, se dirige vers celle du le classeur de sons.

Ce dernier classe le segment selon un critère binaire son/parole par comparaison à des modèles de mixtures de gaussiennes. Il y a 5 classes de sons. Le résultat du traitement est un paramètre booléen qui valide le segment comme n'étant pas un bruit connu et identifiable.

La reconnaissance automatique de la parole parallélise le traitement. Il y a $n+1$ moteurs configurés avec des modèles de langage différents mais partageant le même lexique et les mêmes modèles acoustiques. n correspond au nombre de "dialogues" différents pour le système. Pour chacun d'eux, un modèle de langage fermé est créé à partir de l'ensemble des formulations possibles pour les commandes disponibles. Le $n+1$ème moteur de reconnaissance est configuré avec un modèle de langage large vocabulaire appris sur le corpus CGN: c'est le modèle général (par opposition à ceux des moteurs spécifiques). Chaque moteur produit, par segment, une hypothèse de transcription tandis que le module général en produit trois (mes trois meilleures).

Toutes ces hypothèses passent ensuite un test de similarité, i.e. chaque hypothèse est comparée aux 3 hypothèses issues du moteur général et la distance de Levenstein normalisée par le nombre de mots établit la mesure de similarité. Selon si cette distance est supérieure ou inférieure au seuil expérimentalement défini, l'hypothèse est validée ou rejetée. Dans le cas où plus d'une hypothèse est valide, la distance la plus courte ou, le cas échéant, une sélection aléatoire, décide de l'unique transcription qui est le résultat du test de similarité. Si aucune hypothèse ne passe le test, le segment original est rejeté.

En dernier lieu, le résultat du test de similarité est filtré par la valeur du booléen produit par le classeur de sons pour le même segment initial. Si un segment a été associé à une classe de son, même si la branche de reconnaissance de la parole produit une transcription, celle-ci est rejetée.

Deux mécanismes supplémentaires, pour améliorer la fiabilité de l'analyse sonore, ont été ajoutées.

La première consiste en l'utilisation d'un niveau d'attention ajusté en fonction de la détection d'un mot clé combiné à la progression dans le dialogue.

En second, une adaptation spécifique à l'utilisateur cible est appliquée. Dix phrases phonétiquement équilibrées sont enregistrées par l'utilisateur servant à ajuster les paramètres des modèles acoustiques partagés selon la méthode Maximum Likelihood Linear Regression [70].

8.4.4 Evaluation

Rappelons les deux axes d'amélioration envisagés:

- Fiabilité de la reconnaissance
- Estimation de l'attention

La fiabilité est optimisée de par l'adaptation acoustique au locuteur, la validation ou le rejet des hypothèses avec la reconnaissance parallèle, le classement des sons et le test de similarité. Ce denier test permet trois actions:

- Confirmer une hypothèse correcte
- Rejeter une hypothèse incorrecte
- Corriger une hypothèse partiellement correcte

Du côté de l'attention, celle-ci est gérée par, dans l'ordre, la détection d'un mot clé, une progression significative dans le dialogue, la reconnaissance parallèle et le test de similarité.

Le système de traitement audio a été testé sur des données collectées dans les conditions réelles de déploiement, i.e. enregistrées dans la maison cible. Cinq utilisateurs ont été enregistrés, chacun prononçant 58 énoncés: 10 phrases phonétiquement équilibrées pour l'adaptation acoustique, 20 phrases incluses dans les commandes disponibles, 22 phrases en dehors du périmètre des dialogues et 6 phrases qui constituent des commandes disponibles auxquelles on enlève un ou deux mots de manière aléatoire. Les résultats de la première phase de test sont montrés dans les tables 8.1 à 8.4.

System	Recognition rate	False-positive rate
Baseline + adaptation	15%	0%
Baseline + adaptation + similarity test	85%	0%

Table 8.1: Taux de validation pour les commandes du système

System	Recognition rate	False-positive rate
Baseline + adaptation	9.09%	0%
Baseline + adaptation + similarity test	0%	0%

Table 8.2: Taux de validation pour les commandes non incluses dans le système

System	Recognition rate	False-positive rate
Baseline + adaptation	16.67%	0%
Baseline + adaptation + similarity test	66.67%	0%

Table 8.3: Taux de validation pour les commandes partielles du système

Dans une seconde phase du système, la robustesse au bruit a été mise à l'épreuve. Divers types de bruits environnants ont été joués en même temps que les segments vocaux à traiter.

Noise type	Recognition rate	False-positive rate
Washing machine	74%	11%
Dutch speaker	53%	11%
Music	47%	5%
Crowd	42%	11%

Table 8.4: Taux de validation pour les commandes du système

Noise type	Recognition rate	False-positive rate
Washing machine	0%	0%
Dutch speaker	0%	0%
Music	0%	0%
Crowd	0%	3.64%

Table 8.5: Taux de validation pour les commandes non incluses dans le système

Noise type	Recognition rate	False-positive rate
Washing machine	40%	0%
Dutch speaker	60%	0%
Music	20%	0%
Crowd	60%	0%

Table 8.6: Taux de validation pour les commandes partielles du système

Le système atteint un taux de 85% de reconnaissance dans un environnement propre, ne donne aucun faux positifs et corrige 66% des commandes partielles. En environnement bruité, ces taux se dégrade.

Plus de détails sur le protocole d'évaluation et les résultats obtenus sont inclus dans le document de thèse.

8.5 Compréhension du langage appliqué au dialogue vocal

8.5.1 Introduction

On parle de Compréhension Du Langage (CDL) lorsque l'objectif est d'extraire une signification, quelle qu'elle soit, d'une phrase ou d'un texte (considérons ici qu'il est écrit) pour le rendre interprétable par un programme informatique. Le domaine s'est développé récemment du fait des progrès, notamment, de la reconnaissance de la parole.

Pour un SDV, la CDL est l'interface uni-directionelle qui convertit les mots de l'utilisateur tels que hypothétisés par le module de reconnaissance automatique en une ou plusieurs unités d'action du dialogue. Ces dernières sont en

quantité limitée et dépendantes de l'état, à chaque tour, du dialogue contrôlé par le gestionnaire.

La langue naturelle parlée s'inscrit dans un espace complexe et immense dans lequel les combinaisons de mots, les références aux concepts passés, les références culturelles, les figures de style, etc, ajoutent des niveaux de traitement entrelacés.

Un système de CDL remplit plusieurs conditions. Il doit:

- Autoriser les variations dans l'expression des intentions de l'utilisateur. Contrairement aux premiers systèmes qui se basaient sur des commandes prédéfinies, un système qui prétend utiliser la langue naturelle doit accepter des expressions dérivées des données initiales.

- Maintenir l'initiative mixte. L'initiative définit le degré avec lequel une entité domine dans la direction du dialogue. Une initiative système impose un suivi strict des contraintes du modèle et réduit fortement la spontanéité tandis qu'une initiative utilisateur ôte le bénéfice de la gestion du dialogue puisque la machine réagit tour après tour sans guider l'utilisateur. Une initiative mixte fragilise le système de CDL mais modélise mieux les interactions humaines.

- Intégrer les éléments de contexte locaux du dialogue. L'intérêt du dialogue tient dans le traitement de l'information selon un modèle qui s'étend sur plusieurs tours d'interaction. Les informations échangées lors des tours précédents peuvent modifier ou ajouter un sens au tour courant. De fait, un système de CDL se doit de les intégrer à bon escient dans l'analyse effectuée.

- Intégrer les éléments de contexte partagés et les paramètres de l'environnement. Il est bien entendu que nous supposons que certains faits sont connus par notre interlocuteur lorsque nous engageons la conversation. Cela permet d'optimiser l'échange d'information et de susciter un intérêt. Un module de CDL qui intégre une culture, un savoir, une identité et des notions de l'environnement dans lequel il se situe permet d'atteindre un niveau de naturel bien plus élevé qu'un système amnésique et inculte.

Basé sur ces prérequis, nous proposons un système de CDL dont les modules s'organisent comme sur la figure 8.4.

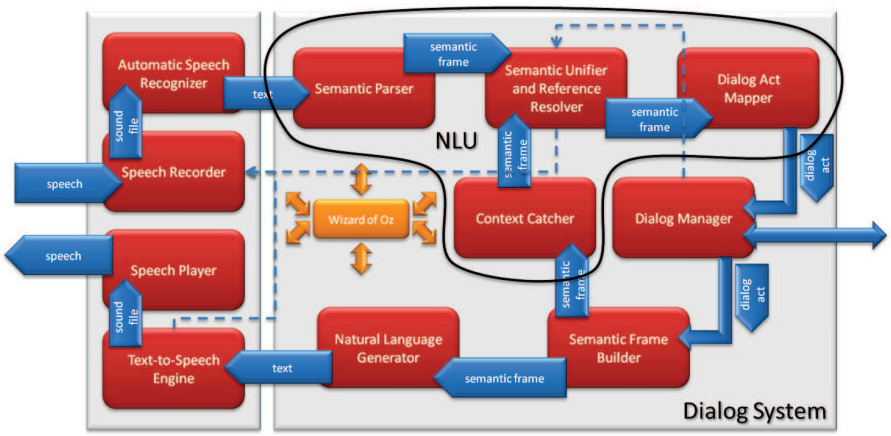


Figure 8.4: Architecture du sous-système de compréhension du langage

Rappelons que des hypothèses (7 au maximum) de transcription de segments de texte, correspondant chacune à un tour de parole, sont soumises au sous-système de CDL. En sortie, des actes de dialogue, correspondant à l'impact que de telles expressions ont sur le dialogue dans son état actuel, sont produites. Figure 8.5 montre la séquence de traitement.

Figure 8.5: Séquence de traitement du langage

Dans les sections suivantes, nous visiterons les modules de traitement et tenteront de répondre aux quatre conditions qui font du système un système complet.

8.5.2 Extraire les concepts sémantiques du texte

La langue naturelle évolue perpétuellement et peut être considérée comme un ensemble infini bien que composé d'un vocabulaire fini (difficilement dénombrable). Extraire des actes de dialogue paramétriques directement à partir du texte demande une transcription parfaite du signal de parole et d'avoir listé toutes les manières de formuler tous les actes de dialogue, ce qui rend la tâche virtuellement impossible.

Dans un premier temps, nous proposons de travailler dans un espace sémantique. La sémantique n'étudie pas la forme des mots mais bien leur signification et celle de leurs combinaisons. Cette signification à plusieurs niveaux.

Un analyseur sémantique, dans notre cas, cherche à extraire des étiquettes sémantiques à partir des mots. Le résultat prend la forme d'une trame sémantique qui est constituée d'un but, c'est l'intention générale du segment, et de zéro ou plusieurs paramètre(s) qui sont des paires (identifiant+valeur) précisant ce but. L'analyseur sémantique base son traitement sur des règles de transformation ordonnées qui sont appliquées une à une dès lors que leurs conditions d'application sont respectées [23, 92, 95].

Ces règles sont apprises sur un corpus annoté. L'algorithme d'apprentissage teste et cherche les meilleures règles parmi toutes celles possibles et transforme le corpus initial jusqu'à obtenir le corpus de référence ou ne plus pouvoir appliquer de transformation qui produirait une amélioration. Pour ne pas perdre en variabilité, il est important de fournir un corpus d'apprentissage aussi complet que possible.

Cette étape projette l'information à traiter vers un espace sémantique réduit et fermé. L'analyse par transformations successives permet de détecter des sous segments de phrases comme sources de concepts sémantiques, de combiner les concepts entre eux et de les augmenter itérativement. De plus, les conditions d'applicabilité des règles de transformation utilisent des motifs dans l'énoncé ce qui est bien adapté à la langue parlée, plus flexible sur les règles de syntaxes [51, 187, 199, 200], et peut passer outre des erreurs de reconnaissance introduites par le module précédent.

8.5.3 Résoudre les références locales

Dans une trame sémantique issue de l'analyse par transformations successives, certains paramètres peuvent être relatifs au contexte du dialogue, c'est à dire correspondre à un concept mentionné précédemment par l'utilisateur ou le système. Ces concepts doivent être mémorisés au cours du dialogue pour être utilisés à bon escient lorsque l'utilisateur y fait référence.

Ici, nous nous intéressons aux références issues du dialogue qui peuvent donc être retrouvées dans l'historique de ce dernier. Le design du sous-système de CDL permet d'accéder à deux sources de contexte du dialogue: le Context Catcher (CC) met à jour la CDL à partir des actes de dialogue générés par le gestionnaire et le Dialog Act Mapper (DAM) récupère l'ensemble des actes de dialogue et leurs paramètres à partir du gestionnaire.

Ces deux sources sont intégrées aux trames sémantiques, si nécessaire, par le module du SURR.

Le SURR est basé sur une forêt, i.e. un ensemble d'arbres dont les noeuds sont des trames sémantiques partiellement définies (voir figure 8.6). Chaque noeud a donc un but associé, qui peut être indéfini, et zéro ou plusieurs paramètres dont le nom comme la valeur peut être libre.

Une trame sémantique à traiter est placée dans les arbres, soit complètement, soit en partageant ses paramètres et en copiant son but. L'algorithme de résolution cherche un chemin vers un noeud ou un ensemble de noeuds caractérisés comme étant "racines". Ceci n'implique pas qu'ils soient à la base des

arbres: les bases des arbres sont des racines mais les racines ne sont pas toutes des bases.

Ces arbres forment donc une taxinomie dynamique des concepts sémantiques dont les branches sont modifiables par le CC en fonction des actes de dialogue produit par le gestionnaire du dialogue.

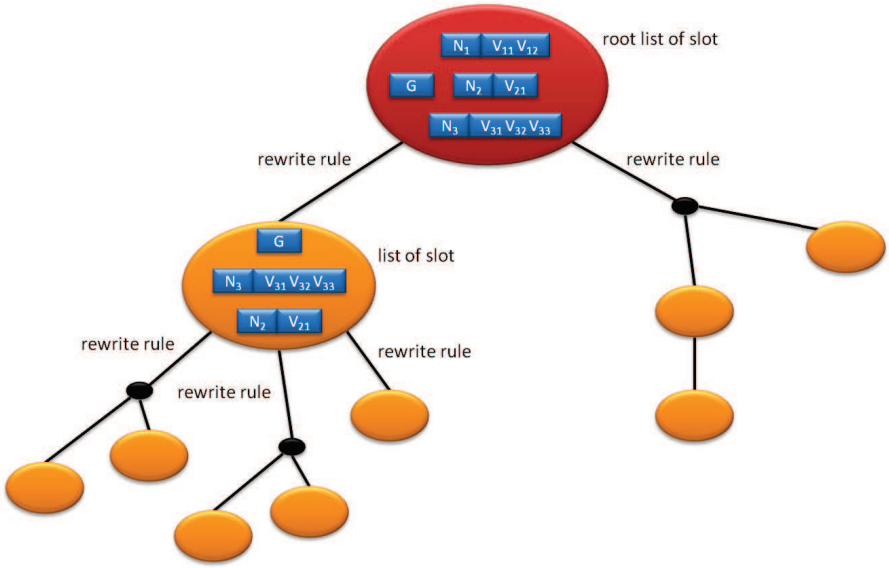


Figure 8.6: Structure interne du SURR

Étudions deux exemples, l'un pour illustrer la résolution d'une référence locale par l'action du CC et l'autre pour illustrer les effets du DAM.

Prenons le dialogue suivant:

- Utilisateur: "Bonjour, je voudrais commander des pizzas"
- Système: "Oui, combien de pizzas voulez-vous?"
- Utilisateur: "Deux"
- Système: "D'accord, quelles recettes?"
- ...

Le 2ème tour, dans lequel l'utilisateur fait référence au nombre de pizzas n'est interprétable en lui-même que par le fait qu'un lien se crée entre le nombre deux, la quantité deux et la quantité de pizzas. Pour ce faire, il faut créer un noeud racine dont le but est indéfini et dont l'unique paramètre est la quantité de pizzas et relier ce noeud à celui contenant un but indéfini et un nombre comme paramètre. Ainsi, le contexte local est intégré dynamiquement grâce au CC qui ajoute/détruit noeuds et liens.

Continuons le dialogue.

- ...
- Utilisateur: “Les recettes 4 fromages et napolitaines”
- Système: “D’accord, vous désirez deux pizzas: une 4 fromages et une napolitaine, c’est correct?”
- Utilisateur: “Oui”
- Système: “Vos pizzas seront prêtes dans 20 minutes. Il vous faut autre chose?”
- Utilisateur: “Non, merci”
- Système: “Au revoir”

Ici, le système demande à l’utilisateur de confirmer la commande. Une confirmation courte, positive est exprimée par l’utilisateur, qu’il faut relier à l’état actuel du dialogue.

Cette fois, on utilisera le DAM. Celui-ci établit, à partir de l’état actuel du dialogue maintenu par le gestionnaire, quels sont les actes de dialogue disponibles et relie ceux-ci aux trames sémantiques reçues.

Dans le 4ème tour, l’acte de dialogue qui confirme, i.e. qui donne la valeur de la confirmation, est unique bien qu’il puisse apparaître à d’autres endroits du dialogue (5ème tour). La confirmation est donc contextualisé par le mapper qui déclenche l’acte de dialogue correspondant à l’état actuel du dialogue et non en réponse à la question: “Il vous faut autre chose?”

8.5.4 Résoudre les références communes et situer l’interaction

Un interlocuteur humain détient et utilise, lorsqu’il communique un savoir, des croyances partagées et des codes sociaux qui influent sur le contenu d’une conversation avec un autre humain. Un système de dialogue, implémenté sur une machine, accéde aux APIs, functions, librairies, capteurs et ressources internet de son hôte.

Ces dernières sources d’information, infinies, pour être utiles, doivent être intégrées au bon endroit, au bon moment. Elles permettent de résoudre des références quant à la situation géographique, temporelle, les conditions climatiques, l’économie et de profiler l’utilisateur.

Nous utilisons le SURR pour ces inclusions. Tout comme les ajouts du CC, des appels aux sources externes d’information sont des noeuds virtuellement instanciés en permanence qui, lorsque nécessaire à la résolution de références, exécutent leur fonctionnalité cible.

Appuyons nous, ici encore, sur un exemple. Le terme “aujourd’hui” réfère à la date du jour, un concept partagé par tous les humains respectant le même calendrier. Il est simple pour un système d’implémenter une fonction d’obtention

de cette date selon le format souhaité. C'est le rôle d'un noeud du SURR qui contient un paramètre "date du jour" constamment porteur d'une valeur. Lorsque le concept "aujourd'hui" est extrait d'un énoncé, le chemin vers un noeud racine passe par le remplacement de "aujourd'hui" avec la valeur du paramètre "date du jour", transformant un concept relatif en un concept à valeur absolue avant d'être transmis au DAM.

Le même principe s'applique pour le jour de la semaine, les conditions météorologiques, l'heure, la localisation du système, etc... Aisément, toute source d'information utile au système de dialogue vocal visé est intégrable par le biais d'un module malléable qu'est le SURR et indépendamment des autres composants.

8.5.5 Unifier les espaces sémantiques

Un dernier pas vers la simplification et la modularité du système de compréhension du langage, effectif au niveau du SURR, tient dans l'unification des espaces sémantiques.

Il a été mentionné précédemment que l'analyseur sémantique extrayait les concepts au plus bas niveau possible, c'est à dire au niveau quasi-primitif de la taxinomie maintenue au coeur du SURR. Ce constat s'appuie sur le fait que seuls les informations portées par l'hypothèse de transcription sont utilisées puisque les éléments de contexte interviennent plus tard dans le traitement. Le DAM, quant à lui, agit dans un espace sémantique dynamique restreint.

De fait, il est nécessaire de projeter les concepts sémantiques extraits par l'analyseur sémantique vers les concepts définis par le gestionnaire de dialogue. Ce traitement est effectué par le SURR,. Les noeuds racines correspondent aux concepts acceptables pour le DAM. Cette unification rend indépendant le développement de l'analyseur et du gestionnaire, ce qui, en conséquence, rend leur remplacement et leur modification aisés et rapides.

8.5.6 Joindre les attentes du gestionnaire dialogue

Une fois la trame sémantique extraite de l'énoncé, contextualisée et unifiée, il reste à trouver la correspondance entre celle-ci et un acte de dialogue paramétrique.

Nous avons vu que le DAM récupère la liste des actes de dialogue disponibles selon l'état du dialogue, plus de détails sont inclus dans le document de thèse. Ainsi, il établit un lien direct entre l'intention d'un acte de dialogue et le but d'une trame sémantique. Le rôle de cet ultime composant de la compréhension du langage est donc d'instancier l'acte de dialogue correspondant et, dans le cas où il n'y a pas de lien existant, de rejeter l'entrée de l'utilisateur, ce qui déclenche les mécanismes de récupération des erreurs.

8.5.7 Conclusion

Le sous-système de compréhension du langage utilise pleinement les aspects de séquencement pour traiter les énoncés en entrée. Son fonctionnement à l'échelle

du composant permet le remplacement de chacun d'eux, la création rapide de nouveaux dialogues et de services ou l'extension des capacités de SDV.

8.6 Modélisation des dialogues: linked-form filling

8.6.1 Introduction

Dès lors que l'interaction vocale entre un humain et une machine excède un tour, on parle de dialogue. Le terme sous entend que l'entité non-humaine est capable d'établir une relation entre les tours d'interaction, i.e. elle maintient un état du dialogue lui permettant de guider un utilisateur dans les services proposés [88]. De fait, un système de dialogue vocal nécessite un organe de prise de décision et de maintien de l'état du dialogue pour contrôler son comportement. Figure 8.7 résume le rôle qu'endosse un tel composant.

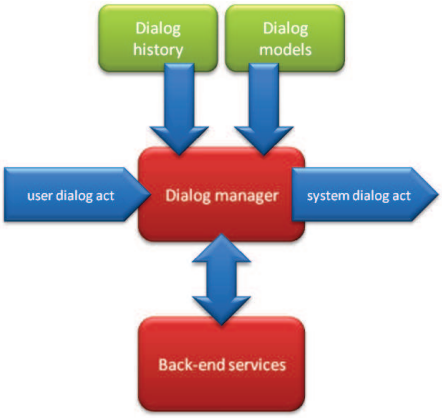


Figure 8.7: Le rôle du gestionnaire de dialogue

La gestion du dialogue est rendue difficile de par plusieurs phénomènes linguistiques [67]. Naturellement, l'esprit humain intègre l'information exprimée par un locuteur de manière continue. Un système de dialogue procède itérativement et requiert une représentation formelle des éléments informatifs. En conséquence, il faut non seulement déterminer quels sont ces éléments, les extraire du signal capté mais également segmenter le flux de parole en unités d'action du dialogue.

L'observation de conversations entre humains a montré l'existence de phénomènes de retour de l'auditeur pour maintenir la compréhension mutuelle (back channel en anglais).

En plus de ces informations non utiles parce que non prises en compte par le système de dialogue, le langage parlé introduit des hésitations, des mots superficiels pouvant être attribués à des tics de langage ou des expressions ambiguës.

Enfin, nous l'avons introduit précédemment, les ellipses (omissions de mots), les indirections ou encore les anaphores complexifient la tâche d'interprétation.

Pour gérer le dialogue, plusieurs méthodes ont été appliquées. Dès 1966, la notion de paires adjacentes est proposée [204]. Cette analyse porte sur la reconnaissance de motifs dans le discours. Elle est locale, i.e. elle ne porte que sur le contenu du tour courant et non sur un historique de dialogue et dans un objectif à long terme.

En 2000, Larsson et Traum proposent de modéliser l'état du dialogue par le cumul de l'information échangée: c'est l'Information State [101, 134]. Celui-ci est mis à jour par des règles conditionnées par l'état actuel du dialogue et l'acte de dialogue le plus récent de l'utilisateur. En plus, une stratégie de mise à jour permet de résoudre les ambiguïtés dans la sélection de la meilleure règle à appliquer. Cette méthode est bien adaptée aux agents conversationnels puisqu'elle ne requiert pas de définition de mesure d'objectif et de performance. Ainsi, elle étend le formalisme des paires adjacentes du fait de la conservation d'un historique de dialogue et la sélection d'une réplique selon celui-ci. La qualité de l'interaction, cependant, est strictement attribuable aux concepteurs qui définissent quels sont les éléments de l'information utiles au dialogue, leur représentation formelle, les règles de mises à jour et la stratégie de sélection.

Un modèle hybride, basé sur les mêmes principes de représentation et de logique a été proposé par Lison [115, 116, 117]. Un règle de mise à jour définit une multitude d'effets auxquels est associée une probabilité pour chacun d'eux. La valeur de ces éléments stochastiques est apprise sur des données collectées préalablement. L'état du dialogue est modélisé par un réseau bayésien dynamique régit par les règles de mise à jour probabilistes. Le gestionnaire de dialogue, en appliquant ces dernières, modifie les variables de l'état courant et décide de l'action la plus appropriée à exécuter.

Lee et al. on adopté un formalisme hybride entre l'Information State et la modélisation par apprentissage automatique [91, 106, 108, 109]. Le modèle est fait d'une base de données d'exemples, appris sur un corpus annoté de dialogue. Les annotations sont multi-niveaux et définissent les actes de dialogue primaires, les actes de discours et un vecteur d'historique de dialogue. Chaque tour de parole dans le corpus est appelé une situation. L'algorithme cherche à rapprocher un énoncé de l'utilisateur, dont l'annotation est produite automatiquement par des composants de la compréhension du langage, à une entrée de la base d'exemples. La correspondance la plus proche définit quelle action doit être exécutée. Elle correspond à l'action prise dans le dialogue sur lequel la base d'exemples a été apprise . La comparaison est effectuée par un mécanisme classique de recherche dans les bases de données et de relâchement/addition de contraintes. Il y a un avantage certain à utiliser ce formalisme puisqu'il ne demande pas de définir précisément le domaine, l'ensemble des actions possibles pour le système et leur conditions d'applicabilité. Cependant, il n'exclut pas l'enregistrement des dialogues et la lourde tâche qu'est l'annotation des données. Certaines situations, qui n'apparaissent pas pendant l'apprentissage, sont rattachées aux exemples plus ou moins similaires, ce qui limites les possibilités de dialogue.

De nombreuses applications de dialogue vocal basent la gestion sur des automates finis. Les états du dialogue n'encode pas explicitement l'historique du dialogue mais les transitions entre ceux-ci définissent un chemin vers un état final. Cette modélisation présente les avantages d'être simple, de limiter les entrées de l'utilisateur et de pouvoir définir, pour chaque état, l'ensemble des commandes disponibles. Dans les inconvénients, on pourra citer l'initiative totalement orientée vers la machine et l'impossibilité pour l'utilisateur de combiner plusieurs éléments d'information dans un tour de parole.

Plus récemment, les modèles stochastiques ont fait leur premiers pas dans la gestion du dialogue. Un Processus de Décision Markovien (PDM) [113, 114, 167] est défini par un ensemble d'états (S), un ensemble d'actions (A) que le système peut faire, un état initial (S_0), une fonction de transition ($P(s'|a, s')$) et une fonction de récompense ($R(s, a)$). La stratégie ($\pi(s)$)définit l'action la plus probable et donc celle qu'un système doit exécuter, dans un état donné de dialogue. Celle-ci est calculée automatiquement à partir d'un ensemble d'apprentissage. Elle a pour objectif d'optimiser l'efficacité du système en maximisant les résultats et en en minimisant le coût (opposé du produit de la fonction de récompense). Les PDMs demandent de collecter des dialogues qui explorent tous les états possibles, ce qui croît rapidement avec le nombre de variables disponibles dans les dialogues. L'avantage en est la facilité de l'apprentissage et le fait que seul le domaine est défini, et non le comportement du gestionnaire. Cela rend l'interaction plus naturelle et offre des possibilités d'apprentissage continue de la stratégie. Il a été montré, cependant, que l'objectif d'un dialogue n'est pas toujours défini et mesurable et que dans ce cas, cette approche devient impossible.

En rajoutant un niveau d'abstraction aux PDMs, dans le but de prendre en compte, dans la prise de décision, l'incertitude inhérente à la transcription et à la compréhension de la parole, Young et al. ont introduit les MDP Partiellement Observables (PDMPOs) [25, 28, 79, 207, 210, 216, 217, 218, 221, 221]. Ces modèles héritent de l'ensemble d'états, l'ensemble d'actions et la fonction de transition des PDMs. Ils y ajoutent un ensemble des observations (O), une fonction d'observation ($P(o|s, a)$) et un belief state ($b(s)$). Cette dernière fonction définit la probabilité, à un instant donné, d'occuper un état du modèle. En effet, ceux-ci ne sont pas observables et ne peuvent être estimer que par les observations qu'ils émettent. La fonction de récompense prend en compte cette incertitude, tout comme la stratégie qui ne prend plus de décision basée sur l'état courant réel mais sur le belief state. L'apprentissage porte également cet accroissement de la complexité et l'explosion combinatoire des possibilités rend la recherche de la stratégie optimale difficile, voire impossible pour des dialogues dont le nombre de variables excède un certain seuil. Plusieurs méthodes pour réduire l'espace de recherche ont été proposées [24, 81, 119, 135, 150, 190, 206, 208, 209]. Des méthodes de simulation de l'utilisateur [31, 32, 54, 93, 111, 146, 166] ont également été implémentées qui permettent d'entraîner un système sans pour autant demander à des utilisateurs d'interagir avec un système sub-optimal. La construction du simulateur est cependant parfois aussi ardue que la définition du modèle de dialogue.

8.6.2 Hiérarchies de tâches pour modéliser le dialogue

La modélisation du dialogue peut également se faire par décomposition des tâches à effectuer et une hiérarchisation de celles-ci [18, 72, 112, 161, 162, 163, 164, 165, 182]. Le principe de base étant: une tâche non primitive s'exécute de par l'exécution de toutes ces tâches filles et une tâche primitive s'exécute par elle même. La conception consiste alors à définir les tâches de dialogue primitives et à les agencer dans un modèle hiérarchique de façon à ce qu'un dialogue corresponde à un noeud de la hiérarchie pour lequel il faut exécuter tous les noeuds de niveaux inférieurs (jusqu'au niveau primitif).

Comme exemple, la figure 8.8 montre comment modéliser un dialogue pour envoyer un message à un correspondant.

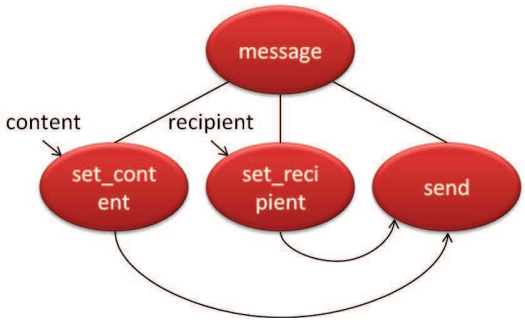


Figure 8.8: Un exemple de hiérarchie de tâches

Les trois tâches primitives sont exécutées par le système qui consulte l'utilisateur pour obtenir les variables d'entrée (content et recipient). Une application du modèle pourrait être:

- Utilisateur: "Bonjour, prépare l'envoi d'un message"
- Système: "Pour qui est le message?"
- Utilisateur: "Paul"
- Système: "Quel en est le contenu?"
- Utilisateur: "Je serai en retard"
- Système: "Le message est envoyé"
- Utilisateur: "Merci"

Ce modèle peut-être étendu facilement en rajoutant des tâches telles que la confirmation du destinataire, le type de message, le sujet, etc... De plus la notion d'extra-information apportée par l'utilisateur, impossible dans un automate fini ou augmentant la complexité dans d'autres formalismes, est directe et

localement influente ici. Cela permet à l'utilisateur de déclarer ces intentions en même temps que de donner le destinataire, ce qui ferait: "Bonjour, prépare l'envoi d'un message pour Paul".

Cependant, la définition d'une hiérarchie, sa maintenance, son extension et son entretien deviennent vite difficiles avec l'augmentation du nombre de variables/concepts ou l'ajout de comportements plus élaborés.

Pour bénéficier des avantages de la modélisation par hiérarchies de tâches tout en excluant certains de ces inconvénients, nous proposons un formalisme que nous appellerons Linked-Form Filling (LFF). Celui-ci est basé sur des formulaires, conditionnés et liés pour spécifier les contraintes du service à interfacer. Ces modèles sont ensuite "compilés" pour produire un modèle de tâche selon le standard ANSI/CEA-2018 [41]. La librairie Disco [161, 165] permet de raisonner sur ces modèles et constitue de fait le coeur de notre implémentation de gestionnaire de dialogue.

Nous verrons ici comment est défini ce nouveau paradigme, comment il est compilé et les avantages, comme les inconvénients apportés. Enfin, une section sera consacré à la recherche des actes de dialogue possibles pour un état du dialogue.

8.6.3 Principes du Linked-Form Filling

L'unité de base du LFF est le formulaire. Celui-ci contient un ensemble de slots auxquels une valeur doit être associée pour compléter le formulaire, des actions à exécuter à certains points du dialogue et des liens vers d'autres (sous-)formulaires. Ces liens sont conditionnés: soit obligatoires (par défaut), soit optionnels, soit ignorés. La différence entre optionnels et ignorés est que le système proposera à l'utilisateur de suivre les premiers tandis que les seconds ne le seront que si l'utilisateur en prend l'initiative (en définissant au moins un slot qu'ils contiennent).

La modélisation par form filling utilise ces mêmes principes à l'exception des sous formulaires liés et dans le fait que, généralement, une action conclut un formulaire. Le LFF permet de définir un semi-ordre dans la succession des slots/actions en regroupant ceux-ci à l'intérieur d'un même formulaire.

Il n'y a pas à proprement parlé de hiérarchie entre les formulaires, on parle plutôt de réseau dont les transitions sont orientées. De plus, un sous-formulaire, une fois complété, retourne le contrôle au formulaire appelant.

Prenons un exemple pour illustrer le LFF. Nous nous basons sur l'exemple précédent du service d'envoi de messages que nous complexifions pour mettre en lumière les apports du LFF. Une confirmation est demandée quant au contenu du message et le type de message (mail ou sms) est configurable par l'utilisateur. Si l'utilisateur n'en fait pas la demande, cependant, le système ignore ce dernier paramètre. La figure 8.9 présente le schéma LFF conçu à partir de ces considérations.

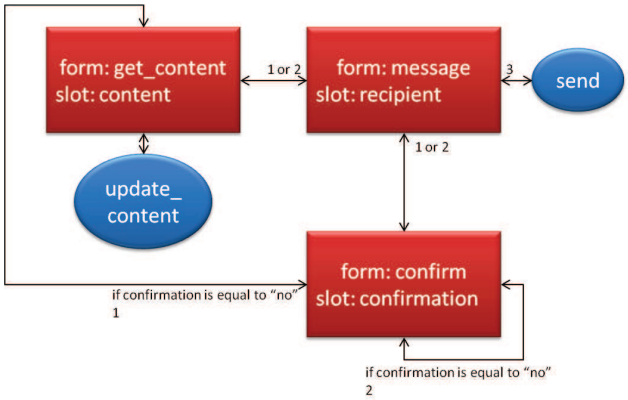


Figure 8.9: Schéma LFF pour le service d'envoi de messages

Dans son déroulement, le modèle demande d'abord de définir le destinataire du message, puis son contenu. Le contenu du message peut être affiché sur un écran en exécutant l'action update_content. Dans le cas ou l'utilisateur ne confirme pas que ce dernier a été correctement transcrit, il peut le redicter, et ce indéfiniment jusqu'à en être satisfait. Le type de message est valué seulement si l'utilisateur l'a mentionné dans un tour portant plus d'information que nécessaire (par exemple: "Le destinataire du sms est Paul" qui informe à la fois le destinataire et le type).

L'avantage de ce modèle en est la clarté et l'intuitivité de conception, cependant, il ne diffère pas réellement de la modélisation par automates finis dans sa flexibilité et l'explosion des combinaisons du dialogue. Ces avantages sont apportés par la transformation en hiérarchie de tâche et la recherche d'actes de dialogue dans le modèle.

La section suivante se penche sur la transformation en modèle hiérarchique.

8.6.4 Transformation en hiérarchie de tâches

Le but de la conversion est de passer d'un modèle LFF dérivé de XML à un modèle conforme au standard ANSI/CEA-2018. Pour ce faire, nous utilisons des règles XSLT récursives qui modifient et créent les éléments du standard.

Toutes les règles ne sont pas expliquées ici mais les principales seront décrites dans leur fonctionnement global. La hiérarchie des éléments LFF est présentée sur la figure 8.10.

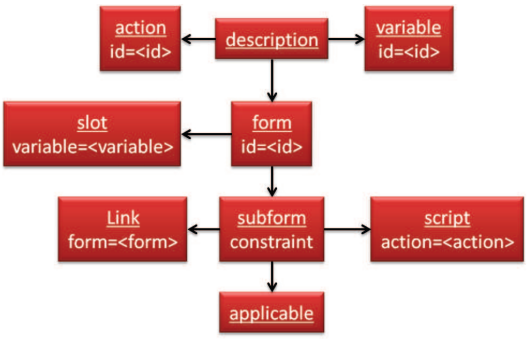


Figure 8.10: Hiérarchie des éléments LFF

Par transformation, les éléments variables et actions produisent des tâches primitives dans le modèle final. Elles seront exécutables par le système seulement. Les variables serviront à enregistrer les valeurs attribuées aux slots et les actions sont des éléments de code externes.

La transformation d'un formulaire produit primitivement une tâche: c'est celle qu'il faut exécuter pour compléter le formulaire. Les règles de conversions entreprennent ensuite une recherche de tous les slots disponibles dans les sous-formulaires descendants et obligatoires. Ce mécanisme permet d'accéder à tous les slots de dialogue en cours dans les tâches parentes et ainsi de garder la possibilité d'extra information. Puis, les formulaires liés sont à leur tour transformés (selon les conditions) pour être ensuite intégrés au modèle.

Tout le processus de transformation est automatique. Bien que le modèle obtenu soit textuel, sa taille étant bien supérieure au modèle original, la lecture par l'humain est difficile. Cependant, toute opération de modification, de maintenance ou de debug est effective dans le modèle LFF, plus accessible, y compris pour un non initié.

8.6.5 Évaluation

Évaluer un nouveau langage de modélisation comprend deux aspects:

- Comment les modèles obtenus se comparent aux modèles construits précédemment?
- Quels sont les avantages (mesurables) du nouveau paradigme par comparaison aux autres?

La première question n'est pas vraiment pertinente. En effet, les modèles LFF sont des biais de conception, ils ne sont pas exécutables en eux-mêmes et doivent d'abord être transformés. De fait, comparer les nouveaux modèles avec les anciens modèles ne dépend que de la qualité de conception de chacun d'eux. Bien que cela prenne plus de temps, il est parfaitement possible de construire les modèles convertis de LFF directement sous le standard ANSI/CEA-2018.

En ce qui concerne les avantages mesurables, nous proposons, dans la table 8.7, de dénombrer les lignes de code et les éléments pour les versions LFF et celles du standard de plusieurs modèles.

	Task hierarchy	LFF	Reduction
Count of lines	139 $[49 - 246]$	30 $[14 - 46]$	76.34% ($\pm$5.26%)
Count of XML elements	87 $[24 - 164]$	18 $[7 - 32]$	76.854% ($\pm$10.184%)

Table 8.7: Comparaison entre les modèles LFF et les hiérarchies de tâches

Cela permet d'estimer la différence de complexité entre les deux méthodes et démontre l'utilité, de ce point de vue, d'un tel outil.

On peut argumenter que nous ne proposons pas ici de nouveau langage mais un outil pour la conception. Un outil s'évalue notamment sur la satisfaction de ses utilisateurs.

Nous cherchons à conduire une étude d'utilisabilité de l'outil auprès d'un groupe aux compétences variés d'utilisateurs. Nous discutons, ici, l'avantage de la conception de modèles non basé sur l'apprentissage automatique (et donc l'utilisation de données) qui permet d'initier une collection de donnée ou de s'en passer complètement.

Dans le cadre de cette thèse, les conditions de développement ont forcées ce choix qui, en fait, est bien plus évident dans le monde industriel [143]. Le coût en termes de temps et d'effort pour collecter un corpus de données suffisant pour apprendre des modèles stochastiques tels que les PDMs ou les POMDPs auprès d'un ensemble d'utilisateurs potentiels est souvent la raison pour laquelle ce formalisme reste aujourd'hui restreint à la communauté de la recherche.

D'un autre côté, le développement de modèles manuellement se doit d'être méticuleux et ne permet pas de partager le modèle avec d'autres concepteurs sans préalablement le documenter et former ces collaborateurs.

L'outil LFF est à la fois visuel et intuitif et, de fait, s'adresse à un plus grand nombre de cibles. De plus, le mapping des tours de parole en langage naturel est facilité par le modèle LFF puisque la décomposition en formulaires et slots délivre une liste exhaustive de ceux-ci.

8.6.6 Conclusion

Les hiérarchies de tâches ont été sélectionnées pour modéliser le dialogue dans la plate-forme développé au cours de cette thèse. Les conditions de développement, l'exigence de prototypage rapide ont influencé ce choix.

Cependant, l'expérience a montré les limites de cette méthode et notamment la difficulté de concevoir, implémenter, debugger, adapter les modèles. Un langage outil, appelé LFF a été proposé qui offre un biais intermédiaire entre le concepteur et le modèle exécutable. Ce paradigme ouvre la conception des modèles aux designers moins expérimentés et facilite celle-ci aux plus avancés.

8.7 Conclusion et perspectives

La principale contribution de cette thèse tient dans la création d'une plate-forme pour le développement de systèmes de dialogue vocal. Celle-ci, libre, possède une modularité qui rend aisée l'implémentation d'un SDV complet, la réutilisation des composants et/ou leur remplacement.

Pour permettre les aspects intelligents, robustes et naturels du système, des mécanismes de détection de l'attention, de correction et de rejet des transcriptions produites par la reconnaissance automatique de la parole ont été proposés. Leur évaluation sur des données collectées dans l'environnement cible a montré leur plus-value et leurs limites. De plus, les méthodes de personnalisation appliquées à la reconnaissance de la parole permettent de fiabiliser les résultats obtenus pour un utilisateur spécifique dans le cadre d'un usage personnel.

Le sous-système de compréhension du langage, séquentiellement, parse partiellement les entrées, résout les références à partir d'une base de connaissance statique et de prédicats dynamiques et sélectionne l'acte de dialogue correspondant. Les cas de rejet déclenchent un protocole de guidage de l'utilisateur gradué et préserve l'intégrité du système. Les mesures effectuées sur des éléments isolés et l'ensemble du système ont montré à quel point ce sous-système était fiable et rapidement configurable.

Enfin, la modélisation sous forme de hiérarchies de tâches, bien que sous-optimale en terme de vitesse de conception, s'avère essentielle pour initier la collection de données nécessaire à un futur système stochastique ou, en terme général, modéliser les scénarios possibles à partir des contraintes du ou des services visés.

A court terme, cette plate-forme pourrait être augmenté avec l'ajout de paraphrases dans le corpus d'apprentissage de l'analyseur sémantique. étendant ainsi les possibilités d'interaction. Une paraphrase constitue une formulation différente, dans le vocabulaire et/ou la grammaire, d'une idée similaire, i.e. qui a la même annotation sémantique. Agrandir le set d'apprentissage, pour les services déjà implémentés, constitue une amélioration de la spontanéité de l'interaction avec le système.

Bien que quelques méthodes de récupération et de prévention des erreurs aient été implémentées, l'état de l'art du domaine pointe certaines lacunes. Si le système s'y prête, il serait profitable d'implémenter ces mécanismes.

Pour conclure, la segmentation des tours de parole pourrait également être assouplie. Le contrôle rigide actuel s'avère fragile et est gênant pour l'utilisateur et l'idée qu'il pourrait avoir quant à la spontanéité du système . Il n'existe pas de synchronisation entre la sortie du système et son entrée. C'est notamment dû à l'architecture client-serveur. Le client devrait intégrer quelques outils de contrôle pour robustiser et filtrer les entrées de l'utilisateur. La CDL incrémentale est, à terme, le mode de fonctionnement d'un SDV puisqu'une conversation humain-humain est régie selon ce principe.

Sur le long terme, je souhaite que la plate-forme puisse être utilisée et modifiée selon les besoins de chacun et ainsi la faire évoluer. Il est difficile de demander à des non-initiés de tester celle-ci étant donné la quantité de tâches qu'il

faut entreprendre. De fait, il n'existe, à ce jour, pas de retour sur les aspects de conception appliqué à l'ensemble du système, ce qui est dommage.

Bibliography

[1] James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Towards Conversational Human-Computer Interaction. *AI Magazine*, 22(4):27–38, 2001.

[2] Jens Allwood, Loredana Cerrato, Kristiina Jokinen, Costanza Navarretta, and Patrizia Paggio. The MUMIN coding scheme for the annotation of feedback, turn management and sequencing phenomena. *International Conference on Language Resources and Evaluation*, 41(3-4):273–287, 2007.

[3] Jens Allwood, David Traum, and Kristiina Jokinen. Cooperation, dialogue and ethics. *International Journal of Human-Computer Studies*, 53(6):871–914, 2000.

[4] Dimitra Anastasiou, Kristiina Jokinen, and Graham Wilcock. Evaluation of WikiTalk–User Studies of Human-Robot Interaction. In *International Conference on Human-Computer Interaction*, pages 32–42, 2013.

[5] M. A. Anusuya and S. K. Katti. Front end analysis of speech recognition: a review. *International Journal of Speech Technology*, 14(2):99–145, 2011.

[6] Harald Aust, Martin Oerder, Frank Seide, and Volker Steinbiss. The Philips automatic train timetable information system. *Speech Communication*, 17(3-4):249–262, 1995.

[7] Janet M. Baker, Li Deng, James R. Glass, Sanjeev Khudanpur, Chin-hui Lee, Nelson Morgan, and Douglas O'Shaughnessy. Research Developments and Directions in Speech Recognition and Understanding, Part 1. *IEEE Signal Processing Magazine*, 26(3):75–80, 2009.

[8] Janet M. Baker, Li Deng, Sanjeev Khudanpur, Chin-hui Lee, James R. Glass, and Nelson Morgan. Historical development and future directions in speech recognition and understanding. *MINDS Report of the Speech Understanding Working Group, NIST*, 2006.

[9] Madeleine Bates, Robert Bobrow, Pascale Fung, Robert Ingria, Francis Kubala, John Makhboul, Long Nguyen, Richard Schwartz, and David

Stallard. Design and performance of HARC, the BBN spoken language understanding system. In *Conference on Spoken Language Processing*, pages 241–244, 1992.

[10] Madeleine Bates, Robert Bobrow, Pascale Fung, Robert Ingria, Francis Kubala, John Makhboul, Long Nguyen, Richard Schwartz, and David Stallard. The BBN/HARC spoken language understanding system. In *International Conference on Acoustics, Speech and Signal Processing*, pages 111–114, 1993.

[11] Frédéric Béchet, Christian Raymond, Frédéric Duvert, and Renato De Mori. Frame Based Interpretation of Conversational Speech. In *Spoken Language Technology Workshop*, pages 401–406, 2010.

[12] Christiane Beuschel, Wolfgang Minker, and Dirk Bühler. Hidden markov modeling for semantic analysis—On the combination of different decoding strategies. *International Journal of Speech Technology*, 8(3):295–305, 2005.

[13] Alan W. Black, Susanne Burger, Alistair Conkie, Helen Hastie, Simon Keizer, Oliver Lemon, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, Jason D. Williams, Kai Yu, Steve Young, and Maxine Eskenazi. Spoken dialog challenge 2010: Comparison of live and control test results. In *SIGdial Meeting on Discourse and Dialogue*, pages 2–7, 2011.

[14] Alan W. Black and Maxine Eskenazi. Real users and real dialog systems : the hard challenge for SDS. In *International Workshop on Spoken Dialog Systems*, pages 29–36, 2012.

[15] Dan Bohus. *Error awareness and recovery in conversational spoken language interfaces*. PhD thesis, Carnegie Mellon Univeristy, 2007.

[16] Dan Bohus, Sergio Grau Puerto, David Huggins-daines, Venkatesh Keri, Gopala Krishna, Rohit Kumar, Antoine Raux, and Stefanie Tomko. Conquest – an Open-Source Dialog System for Conferences. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 9–12, 2007.

[17] Dan Bohus, Antoine Raux, Thomas K Harris, Maxine Eskenazi, and Alexander I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In *Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, pages 32–39, 2007.

[18] Dan Bohus and Alexander I. Rudnicky. RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Eurospeech*, 2003.

[19] Dan Bohus and Alexander I. Rudnicky. Sorry, I didn't catch that! - An Inverstigation of Non-understanding Errors and Recovery Strategies. In *SIGdial Meeting on Discourse and Dialogue*, pages 123–154, 2005.

[20] Dan Bohus and Alexander I. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361, 2009.

[21] Hélène Bonneau-Maynard, Matthieu Quignard, and Alexandre Denis. MEDIA: a semantically annotated corpus of task oriented dialogs in French. In *International Conference on Language Resources and Evaluation*, pages 329–354, 2009.

[22] Philippe Bretier and David Sadek. A Rational Agent as the Kernel of a Cooperative Spoken Dialogue System : Implementing a Logical Theory of Interaction. In *European Conference on Artificial Intelligence*, pages 189–203, 1997.

[23] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–565, 1995.

[24] Trung H. Bui, Mannes Poel, Anton Nijholt, and Job Zwiers. A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. *Natural Language Engineering*, 15(2):273–307, 2008.

[25] Trung H. Bui, Boris Van Schooten, and Dennis Hofs. Practical dialogue manager development using POMDPs. *SIGdial Meeting on Discourse and Dialogue*, pages 215–218, 2007.

[26] Ronnie Cann, Ruth Kempson, and Matthew Purver. Context and well-formedness: The dynamics of ellipsis. *Research on Language and Computation*, 5(3):333–358, 2007.

[27] Daniel Regis Sarmento Caon, Asmaa Amehraye, Joseph Razik, Gérard Chollet, Rodrigo V. Andreao, and Chafic Mokbel. Experiments on acoustic model supervised adaptation and evaluation by k-fold cross validation technique. In *International Symposium on I/V Communications and Mobile Network*, pages 1–4, 2010.

[28] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *National Conference on Artificial Intelligence*, pages 1023–1028, 1994.

[29] Roberta Catizone, Anton Setzer, and Yorick Wilks. State of the art in dialogue management. *Deliverable D5*, 2002.

[30] Mauro Cettolo, Anna Corazza, and Renato De Mori. Language portability of a speech understanding system. *Computer Speech & Language*, 12(1):1–21, 1998.

[31] Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. Clustering behaviors of spoken dialogue systems users. *International Conference on Acoustics, Speech and Signal Processing*, pages 4981–4984, 2012.

[32] Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. Co-adaptation in Spoken Dialogue Systems. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 343–353, 2014.

[33] Ciprian Chelba, Dan Bikel, Maria Shugrina, Patrick Nguyen, and Kumar Shankar. Large scale language modeling in automatic speech recognition. Technical report, Google, 2012.

[34] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Annual meeting of the Association for Computational Linguistics*, volume 13, pages 310–318, 1996.

[35] Gérard Chollet, Asmaa Amehraye, Joseph Razik, Leila Zouari, Houssemeddine Khemiri, and Chafic Mokbel. Spoken dialogue in virtual worlds. In *Development of Multimodal Interfaces: Active Listening and Synchrony*, pages 423–443. Springer, 2010.

[36] Grace Chung. Developing a flexible spoken dialog system using simulation. In *Annual Meeting of the Association for Computational Linguistics*, 2004.

[37] Gavin E. Churcher, Eric S. Atwell, and Clive Souter. Dialogue management systems: a survey and overview. Technical report, Unknown Institution, 1997.

[38] Herbert H. Clark. *Using language*. Cambridge University Press, 1996.

[39] Herbert H. Clark and Edward F. Schaefer. Contributing to Discourse. *Cognitive Science*, 13(2):259–294, 1989.

[40] Ron Cole, Lynette Hirschman, Les Atlas, Hynek Hermansky, Patti Price, Mary Beckman, Steve Levinson, Harvey Silverman, Alan Biermann, Kathy McKeown, Judy Spitz, Marcia Bush, Nelson Morgan, Alex Waibel, Mark Clements, David G. Novick, Clifford Weinstein, Jordan Cohen, Mari Ostendorf, Steve Zahorian, Oscar Garcia, Sharon Oviatt, Victor Zue, and Brian Hanson. The challenge of spoken language systems: Research directions for the nineties. *IEEE Transactions on Speech and Audio Processing*, 3(1):1–21, 1995.

[41] Consumer Electronics Association. Task Model Description (CE TASK 1.0), 2008.

[42] Gennaro Cordasco, Marilena Esposito, Francesco Masucci, Maria Teresa Riviello, Anna Esposito, Gérard Chollet, Stephan Schlögl, Pierrick Milhorat, and Gianni Pelosi. Assessing Voice User Interfaces: The vAssist System Prototype. In *Cognitive Infocommunications*, pages 91–96, 2014.

[43] Ann Cralidis and Celia R. Hooper. Normal changes in the speech of older adults. You've still got what it takes, it just takes a little longer! *SIG 15 Perspectives on Gerontology*, 14(2):47–56, 2009.

[44] Adam Csapo, Emer Gilmartin, Jonathan Grizou, Jingguang Han, Raveesh Meena, Dimitra Anastasiou, Kristiina Jokinen, and Graham Wilcock. Multimodal Conversational Interaction with a Humanoid Robot. In *International Conference on Cognitive Infocommuniations*, 2012.

[45] Géraldine Damnati, Frédéric Béchet, and Renato De Mori. First implementation of the LUNA Spoken Language Understanding strategy on a telephone service application. *Intelligent Information Systems*, pages 499–505, 2008.

[46] Steven B. Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics Speech and Signal Processing*, 28(4):357–366, 1980.

[47] Renato De Mori. *Spoken Dialogues with Computers*. Academic Press, 1998.

[48] Renato De Mori, Frédéric Béchet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. Spoken language understanding: A survey. In *Automatic Speech Recognition and Understanding Workshop*, pages 365–376, 2007.

[49] Alexandre Denis, Frédéric Béchet, and Matthieu Quignard. Résolution de la référence dans des dialogues homme-machine: évaluation sur corpus de deux approches symbolique et probabiliste. In *Conférence du Traitement Automatique du Langage Naturel 2013*, pages 261–270, 2007.

[50] Laurence Devillers, Hélène Maynard, Sophie Rosset, Patrick Paroubek, Kevin McTait, Djamel Mostefa, Khalid Choukri, Laurent Charnay, Caroline Bousquet, Nadine Vigouroux, Frédéric Béchet, Laurent Romary, Jean-Yves Antoine, Jeanne Villaneau, Myriam Vergnes, and Jérôme Goulian. The French MEDIA/EVALDA Project: the Evaluation of the Understanding Capability of Spoken Language Dialogue Systems. In *International Conference on Language Resources and Evaluation*, 2004.

[51] Jacques Duchateau, Tom Laureys, Kris Demuynck, and Parick Wambacq. Handling disfluencies in spontaneous language models. *Language and Computers*, 2003.

[52] Frédéric Duvert, Marie-Jean Meurs, Christophe Servan, Frédéric Béchet, Fabrice Lefèvre, and Renato De Mori. Semantic composition process in a speech understanding system. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5029–5032, 2008.

[53] Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review*, 36(1):1–27, 2011.

[54] Wieland Eckert, Esther Levin, and Roberto Pieraccini. User modeling for spoken dialogue system evaluation. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 80–87, 1997.

[55] Layla El Asri, Romain Laroche, and Olivier Pietquin. DINASTI: Dialogues with a Negotiating Appointment Setting Interface. In *International Conference on Language Resources and Evaluation*, 2014.

[56] Arash Eshghi and Oliver Lemon. How domain-general can we be ? Learning incremental Dialogue Systems without Dialogue Acts. In *Workshop on the Semantics and Pragmatics of Dialogue*, 2014.

[57] Maxine Eskenazi, Alan W. Black, Antoine Raux, and Brian Langner. Let's Go Lab : a platform for evaluation of spoken dialog systems with real world users. In *InterSpeech*, 2008.

[58] European Association for Standardizing Information and Communication Systems. 262: ECMAScript language specification, 1999.

[59] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine learning*, 62:41–62, 1998.

[60] Jean E. Fox Tree. The effects of false starts and repetitions on the processing of subsequent words in spontaneous speech. *Journal of Memory and Language*, 34(6):709–738, 1995.

[61] Olivier Galibert, Gabriel Illouz, and Sophie Rosset. Ritel: an open-domain, human-computer dialog system. *InterSpeech*, pages 909–912, 2005.

[62] Sylvain Galliano, Edouard Geoffrois, Guillaume Gravier, Jean-François Bonastre, Djamel Mostefa, and Khalid Choukri. Corpus description of the ESTER evaluation campaign for the rich transcription of French broadcast news. *International Conference on Language Ressources and Evaluation*, pages 315–320, 2006.

[63] Jean-Luc Gauvain and Chin-hui Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, 1994.

[64] Fabrizio Ghigi, Maria Inés Torres, Raquel Justo, and José-Miguel Benedí. Evaluating spoken dialogue models under the interactive pattern recognition framework. In *InterSpeech*, pages 480–484, 2013.

[65] Jonathan Ginzburg. *The interactive stance.* Oxford University Press, 2012.

[66] Diego Giuliani and Matteo Gerosa. Investigating recognition of children's speech. *International Conference on Acoustics, Speech and Signal Processing*, 2003.

[67] James R. Glass. Challenges for spoken dialogue systems. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999.

[68] Allen L. Gorin, Giuseppe Riccardi, and Jeremy H. Wright. How may I help you? *Speech Communication*, 23(1):113–127, 1997.

[69] Horst-Michael Gross, Christof Schröter, Mueller Steffen, Michael Volkhardt, Erik Einhorn, Andreas Bley, Tim Langner, Christian Martin, and Matthias Merten. I'll keep an eye on you: Home robot companion for elderly people with cognitive impairment. In *International Conference on Systems, Man, and Cybernetics*, pages 2481–2488, 2011.

[70] Jonathan E. Hamaker. MLLR: A speaker adaptation technique for LVCSR. Technical report, Mississippi State University, 1999.

[71] Philip Hanson. *A unified representation for dialogue and action in computer games : bridging the gap between talkers and fighters.* PhD thesis, Worcester Polytechnic Institute, 2010.

[72] Philip Hanson and Charles Rich. A non-modal approach to integrating dialogue and action. In *Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

[73] Sunao Hara, Hiromichi Kawanami, Hiroshi Saruwatari, and Kiyohiro Shikano. Development of a toolkit handling multiple speech-oriented guidance agents for mobile applications. In *International Workshop on Spoken Dialog Systems*, pages 195–200, 2012.

[74] Thomas K Harris and Alexander I. Rudnicky. TeamTalk: A platform for multi-human-robot dialog research in coherent real and virtual spaces. In *National Conference on Artificial Intelligence*, 2007.

[75] Jean-Paul Haton, Christophe Cerisara, Dominique Fohr, Yves Laprie, and Kamel Smaïli. *Reconnaissance automatique de la parole - Du signal à son interprétation.* Dunod, 2006.

[76] Yulan He and Steve Young. Hidden vector state model for hierarchical semantic parsing. In *International Conference on Acoustics, Speech and Signal Processing*, 2003.

[77] Yulan He and Steve Young. Semantic processing using the Hidden Vector State model. *Computer Speech & Language*, 19(1):85–106, 2005.

[78] Yulan He and Steve Young. Spoken language understanding using the hidden vector state model. *Speech Communication*, 48(3-4):262–275, 2006.

[79] James Henderson and Oliver Lemon. Mixture model POMDPs for efficient handling of uncertainty in dialogue management. In *Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, pages 73–76, 2008.

[80] Beth Ann Hockey, Oliver Lemon, Ellen Campana, Laura Hiatt, Gregory Aist, James Hieronymus, Alexander Gruenstein, and John Dowding. Targeted Help for Spoken Dialogue Systems: intelligent feedback improves naive users' performance. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 147–154, 2003.

[81] Jesse Hoey and Pascal Poupart. Solving POMDPs with continuous or large discrete observation spaces. In *International Joint Conference on Artificial Intelligence*, pages 1332–1338, 2005.

[82] Dennis Hofs, Rieks Op den Akker, and Anton Nijholt. A generic architecture and dialogue model for multimodal interaction. *Unknown journal*, 2003.

[83] Kate S. Hone and Robert Graham. Towards a tool for the subjective assessment of speech system interfaces (SASSI). *Natural Language Engineering*, 6:287–303, 2000.

[84] Dan Istrate, Eric Castelli, Michel Vacher, Laurent Besacier, and Jean-François Serignat. Information Extraction From Sound for Medical Telemonitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):264–274, 2006.

[85] Bassam Jabaian, Laurent Besacier, and Fabrice Lefèvre. Investigating multiple approaches for SLU portability to a new language. In *InterSpeech*, pages 2502–2505, 2010.

[86] Kristiina Jokinen and Topi Hurtig. User expectations and real experience on a multimodal interactive system. In *InterSpeech*, 2006.

[87] Kristiina Jokinen, Antti Kerminen, Mauri Kaipainen, Tommi Jauhiainen, Graham Wilcock, Markku Turunen, Jaakko Hakulinen, Jukka Kuusisto, and Krista Lagus. Adaptive dialogue systems-interaction with interact. In *SIGdial Meeting on Discourse and Dialogue*, pages 64–73, 2002.

[88] Kristiina Jokinen and Michael McTear. *Spoken Dialogue Systems*, volume 2. Morgan & Claypool Publishers, 2009.

[89] Kristiina Jokinen and Graham Wilcock. Constructive Interaction for Talking about Interesting Topics. In *International Conference on Language Resources and Evaluation*, pages 404–410, 2012.

[90] Biing-Hwang Juang and Lawrence R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[91] Sangkeun Jung, Cheongjae Lee, and Gary Geunbae Lee. Dialog studio: An Example Based Spoken Dialog System Development Workbench. In *International Conference on Spoken Language Processing*, 2006.

[92] Filip Jurčíček, François Mairesse, Milica Gašić, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. Transformation-based Learning for semantic parsing. In *InterSpeech*, pages 2719–2722, 2009.

[93] Filip Jurčíček, Blaise Thomson, and Steve Young. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language*, 26(3):168–192, 2011.

[94] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[95] Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. Learning to transform natural to formal languages. In *National Conference On Artificial Intelligence*, pages 1062–1068, 2005.

[96] Catherine Kobus, Géraldine Damnati, Lionel Delphin-poulat, and Renato De Mori. Conceptual Language Model Design for Spoken Language Understanding. In *InterSpeech*, pages 3465–3468, 2005.

[97] Catherine Kobus, Géraldine Damnati, Lionel Delphin-Poulat, and Renato De Mori. Exploiting semantic relations for a spoken language understanding application. In *International Conference on Spoken Language Processing*, 2006.

[98] Ivana Kruijff-Korbayová, Heriberto Cuayáhuitl, Bernd Kiefer, Marc Schröder, Piero Cosi, Giulio Paci, Giacomo Sommavilla, Fabio Tesser, Hichem Sahli, and Georgios Athanasopoulos. Spoken Language Processing in a Conversational System for Child-Robot Interaction. In *Workshop on Child-Computer Interaction*, pages 32–39, 2012.

[99] Lori Lamel, Samir Bennacef, Sophie Rosset, Laurence Devillers, Saliha Foukia, Jean-Jacques Gangolf, and Jean-Luc Gauvain. The LIMSI RailTel System : Field trial of a telephone service for rail travel information. *Speech Communication*, 23:67–82, 1997.

[100] Lori Lamel, Sophie Rosset, Jean-Luc Gauvain, Samir Bennacef, Martine Garnier-rizet, and Bernard Prouts. The LIMSI ARISE system. *Speech Communication*, 31:339–353, 2000.

[101] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural language engineering*, 6:323–340, 1998.

[102] Benjamin Lecouteux, Michel Vacher, and François Portet. Distant speech recognition for home automation: Preliminary experimental results in a smart home. *Conference on Speech Technology and Human-Computer Dialogue*, pages 1–10, 2011.

[103] Benjamin Lecouteux, Michel Vacher, and François Portet. Distant speech recognition in a smart home : Comparison of several multisource ASRs in realistic conditions. *InterSpeech*, pages 2273–2276, 2011.

[104] Akinobu Lee. The Julius book, 2010.

[105] Akinobu Lee and Tatsuya Kawahara. Recent Development of Open-Source Speech Recognition Engine Julius. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 131–137, 2009.

[106] Cheongjae Lee, Sangkeun Jung, Jihyun Eun, Minwoo Jeong, and Gary Geunbae Lee. A situation-based dialogue management using dialogue examples. In *International Conference on Acoustics, Speech and Signal Processing*, pages 69–72, 2006.

[107] Cheongjae Lee, Sangkeun Jung, Kyungduk Kim, Donghyeon Lee, and Gary Geunbae Lee. Recent Approaches to Dialog Management for Spoken Dialog Systems. *Journal of Computing Science and Engineering*, 4(1):1–22, 2010.

[108] Cheongjae Lee, Sangkeun Jung, Donghyeon Lee, and Gary Geunbae Lee. Example-based error recovery strategy for spoken dialog system. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 538–543, 2007.

[109] Cheongjae Lee, Sangkeun Jung, and Gary Geunbae Lee. Robust dialog management with n-best hypotheses using dialog examples and agenda. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 630–637, 2008.

[110] Chin-hui Lee and Jean-Luc Gauvain. Speaker adaptation based on MAP estimation of HMM parameters. In *International Conference on Acoustics, Speech and Signal Processing*, pages 558–561, 1993.

[111] Oliver Lemon and Olivier Pietquin. Machine learning for spoken dialogue systems. In *European Conference on Speech Communication and Technologies*, pages 2685–2688, 2007.

[112] Neal Lesh, Charles Rich, and Candace L Sidner. Using plan recognition in human-computer collaboration. In *International Conference on User Modeling*, 1999.

[113] Esther Levin, Roberto Pieraccini, and Wieland Eckert. Using Markov Decision Process for Learning Dialogue Strategies. In *International Conference on Acoustics, Speech and Signal Processing*, pages 201–204, 1998.

[114] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.

[115] Pierre Lison. Declarative design of spoken dialogue systems with probabilistic rules. In *Workshop on the Semantics and Pragmatics of Dialogue*, pages 32–39, 2012.

[116] Pierre Lison. Towards Dialogue Management in Relational Domains. In *SLTC Workshop on Action, Perception and Language*, 2012.

[117] Pierre Lison. Towards Online Planning for Dialogue Management with Rich Domain Knowledge. In *International Workshop on Spoken Dialog Systems*, pages 111–123, 2014.

[118] John Lyons. *Natural language and universal grammar: essays in linguistic theory*. Cambridge University Press, 1991.

[119] Owen Macindoe, Leslie Pack Kaelbling, and Tomas Lozano-Pérez. POMCoP: belief space planning for sidekicks in cooperative games. *Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.

[120] Joseph Mariani. *Spoken language processing*. ISTE, 2009.

[121] John McDonough and Matthias Wölfel. Distant speech recognition: bridging the gaps. In *Hands-free Speech Communication and Microphne Arrays*, 2008.

[122] Michael McTear. *Spoken dialogue technology: Towards the conversational user interface*. Springer Science & Business Media, 2004.

[123] Marie-Jean Meurs, Frédéric Duvert, Frédéric Béchet, Fabrice Lefèvre, and Renato De Mori. Annotation en Frames Sémantiques du corpus de dialogue MEDIA. In *Conférence du Traitement Automatique du Langage Naturel*, 2008.

[124] Marie-Jean Meurs, Frédéric Duvert, Frédéric Béchet, Fabrice Lefèvre, and Renato De Mori. Semantic Frame Annotation on the French MEDIA corpus. In *International Conference on Language Resources and Evaluation*, 2008.

[125] Pierrick Milhorat, Gérard Chollet, and Jérôme Boudy. Un Système de Dialogue Vocal Comme Agent d'Aide à la Personne. In *Journées d'Etude sur la TéléSanté*, 2014.

[126] Pierrick Milhorat, Dan Istrate, Jérôme Boudy, and Gérard Chollet. Hands-free Speech-sound Interactions At Home. In *European Signal Processing Conference*, pages 1678–1682, 2012.

[127] Pierrick Milhorat, Dan Istrate, Jérôme Boudy, and Gérard Chollet. Interactions Sonores Et Vocales Dans l'Habitat. In *Interactions Langagières pour personnes Agées Dans les habitats Intelligents*, pages 17–30, 2012.

[128] Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, and Jérôme Boudy. Multi-step Natural Language Understanding. In *SIGdial Meeting on Discourse and Dialogue*, pages 157–159, 2013.

[129] Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, and Jérôme Boudy. Un Systeme De Dialogue Vocal Pour Les Seniors: Etudes Et Spécifications. In *Journées d'Etude sur la TéléSanté*, 2013.

[130] Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, and Jérôme Boudy. What If Everyone Could Do It? A Framework For Easier Spoken Dialog System Design. In *Engineering Interactive Computing Systems*, pages 217–222, 2013.

[131] Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, Jérôme Boudy, Anna Esposito, and Gianni Pelosi. Building the Next generation of Personal Digital Assistants. In *ATSIP International Conference on Advanced Technologies for Signal & Image Processing*, pages 458–463, 2014.

[132] Raymond J. Mooney. Learning for Semantic Interpretation: Scaling Up without Dumbing Down. In *Workshop on Learning Language in Logic*, pages 7–14, 1999.

[133] Fabrizio Morbini, Kartik Audhkhasi, Kenji Sagae, Ron Arstein, Dogan Can, Panayiotis Georgiou, Shri Narayanan, Anton Leuski, and David Traum. Which ASR should I choose for my dialogue system? In *SIGdial Meeting on Discourse and Dialogue*, pages 394–403, 2013.

[134] Fabrizio Morbini, David Devault, Kenji Sagae, Jillian Gerten, Angela Nazarian, and David Traum. FLoReS : a forward looking , reward seeking , dialogue manager. In *International Workshop on Spoken Dialog Systems*, 2012.

[135] Kevin P Murphy. A survey of POMDP solution techniques, 2000.

[136] Alexis Nasr, Frédéric Béchet, Jean-François Rey, Benoît Favre, and Joseph Le Roux. MACAON - An NLP Tool Suite for Processing Word Lattices. In *Annual Meeting of the Association for Computational Linguistics*, pages 86–91, 2011.

[137] Anh Nguyen and Wayne Wobcke. An agent-based approach to dialogue management in personal assistants. In *International Conference on Intelligent User Interfaces*, pages 137–144, 2005.

[138] Anh Nguyen and Wayne Wobcke. Extensibility and Reuse in an Agent-Based Dialogue Model. In *Web Intelligence and Intelligent Agent Technology Workshops*, pages 367–371, 2006.

[139] Marie Owens, Peter O'Boyle, J. McMahon, Ji Ming, and F. Jack Smith. A comparison of human and statistical language model performance using missing-word tests. *Language and Speech*, 40(4):377–389, 1997.

[140] Tim Paek and Roberto Pieraccini. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communication*, 50(8):716–729, 2008.

[141] Sathish Pammi, Marcela Charfuelan, and Marc Schröder. Multilingual voice creation toolkit for the MARY TTS platform. *International Conference on Language Resources and Evaluation*, 2010.

[142] Christine Pao, Philipp Schmid, and James R. Glass. Confidence scoring for speech understanding systems. *International Conference on Spoken Language Processing*, 1998.

[143] Roberto Pieraccini and Juan Huerta. Where do we go from here? Research and commercial spoken dialog systems. In *SIGdial Meeting on Discourse and Dialogue*, 2005.

[144] Roberto Pieraccini, Esther Levin, and Chin-hui Lee. Stochastic Representation of Conceptual Structure in the ATIS Task. In *Workshop on Speech and Natural Language*, 1991.

[145] Roberto Pieraccini, Evelyne Tzoukermann, Zakhar Gorelov, Esther Levin, Chin-Hui Lee, and Jean-Luc Gauvain. Progress report on the Chronus system: ATIS benchmark results. In *Workshop on Speech and Natural Language*, 1992.

[146] Olivier Pietquin. Statistical user simulation for spoken dialogue systems: what for, which data, which Future? *Workshop on Future directions and needs in the Spoken Dialog Community*, pages 9–10, 2012.

[147] Olivier Pietquin and Thierry Dutoit. Modélisation d'un Système de Reconnaissance pour l'Apprentissage Automatique de Stratégies de Dialogue Optimales. In *Journées d'Etudes sur la Parole*, pages 281–284, 2002.

[148] Florian Pinault. *Apprentissage par renforcement pour la généralisation des approches automatiques dans la conception des systèmes de dialogue oral*. PhD thesis, Orange Labs, 2011.

[149] Florian Pinault, Fabrice Lefevre, and Renato De Mori. Feature-based summary space for stochastic dialogue modeling with hierarchical semantic frames. In *InterSpeech*, pages 284–287, 2009.

[150] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration : An anytime algorithm for POMDPs. *International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

[151] Janne Pylkkönen and Mikko Kurimo. Improving discriminative training for robust acoustic models in large vocabulary continuous speech recognition. In *InterSpeech*, 2012.

[152] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[153] Lawrence R. Rabiner and Biing-Hwang Juang. An introduction to hidden Markov models. *Acoustics, Speech and Signal Processing Magazine*, 3(1):4–16, 1986.

[154] Mathieu Radenen and Thierry Artieres. Contextual hidden markov models. In *International Conference on Acoustics, Speech and Signal Processing*, 2012.

[155] Mathieu Radenen, Pierrick Milhorat, Thierry Artières, Jérôme Boudy, and Gérard Chollet. Etude des HMM paramétriques pour la reconnaissance de parole en environnement bruité. In *Journées d'Etudes sur la Parole*, 2014.

[156] Antoine Raux. *Flexible Turn-Taking for Spoken Dialog Systems*. PhD thesis, Carnegie Mellon university, 2008.

[157] Antoine Raux, Brian Langner, Alan W. Black, and Maxine Eskenazi. LET'S GO: Improving Spoken Dialog Systems for the Elderly and Non-natives. In *European Conference on Speech Communication and Technology*, 2003.

[158] Antoine Raux, Brian Langner, and Dan Bohus. Let's go public! taking a spoken dialog system to the real world. In *InterSpeech*, 2005.

[159] Manny Rayner, Pierrette Bouillon, Beth Ann Hockey, Nikos Chatzichrisafis, and Marianne Starlander. Comparing rule-based and statistical approaches to speech understanding in a limited domain speech translation system. In *International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 21–29, 2004.

[160] Manny Rayner, BA Hockey, and Frankie James. A compact architecture for dialogue management based on scripts and meta-outputs. In *Workshop on Conversational Systems*, pages 54–60, 2000.

[161] Charles Rich. Building task-based user interfaces with ANSI/CEA-2018. *Computer*, pages 20–27, 2009.

[162] Charles Rich, Neal Lesh, Jeff Rickel, and Andrew Garland. A plug-in architecture for generating collaborative agent responses. *International Joint Conference on Autonomous agents and multiagent systems*, pages 782–789, 2002.

[163] Charles Rich and Candace L Sidner. Collaborative discourse , engagement and always-on relational agents. In *Association for the Advancement of Artificial Intelligence Fall Symposia: Dialog with Robots*, 2010.

[164] Charles Rich and Candace L Sidner. Using collaborative discourse theory to partially automate dialogue tree authoring. In *Intelligent Virtual Agents*, pages 327–340, 2012.

[165] Charles Rich, Candace L. Sidner, and Neal Lesh. COLLAGEN: applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4):15–26, 2001.

[166] Stéphane Rossignol, Olivier Pietquin, and Michel Ianotto. Simulation of the grounding process in spoken dialog systems with Bayesian Networks. In *Spoken Dialogue Systems for Ambient Environments*, pages 110–121. Springer, 2010.

[167] Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialog management for robots. In *Annual Meeting of the Association for Computational Linguistics*, 2000.

[168] David Sadek. Dialogue acts are rational plans. In *The Structure of Multimodal Dialogue, Second VENACO Workshop*, 1991.

[169] David Sadek, A. Ferrieux, A. Cozannet, P. Bretier, F. Panaget, and J. Simonin. Effective human-computer cooperative spoken dialogue: the AGS demonstrator. In *International Conference on Spoken Language Processing*, volume 1, pages 546–549, 1996.

[170] M. D. Sadek, P. Bretier, and F. Panaget. ARTIMIS: Natural dialogue meets rational agency. In *International Joint Conference on Artificial Intelligence*, pages 1030–1035, 1997.

[171] Hugues Sansen, Jean-Louis Baldinger, Jérôme Boudy, Gérard Chollet, Pierrick Milhorat, and Stephan Schlögl. vAssist : Building The Personal Assistant For Dependent People - Helping Dependent People to Cope With Technology Through Speech Interaction. In *HEALTHINF International Conference on Health Informatics*, 2014.

[172] Jeff Sauro and James R Lewis. *Quantifying the user experience: Practical statistics for user research*. Elsevier, 2012.

[173] Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Garret, and Brian Strope. "Your Word is my Command": Google Search by Voice: A Case Study. In *Advances in Speech Recognition*, volume 2, pages 61–90. Springer, 2010.

[174] Stephan Schlögl, Gérard Chollet, Pierrick Milhorat, Jirasri Deslis, Jacques Feldmar, Jérôme Boudy, Markus Garschall, and Manfred Tscheligi. Using Wizard of Oz To Collect Interaction Data For Voice Controlled Home Care And Communication Services. In *Signal Processing, Pattern Recognition and Applications*, pages 12–14, 2013.

[175] Stephan Schlögl, Pierrick Milhorat, and Gérard Chollet. Designing , Building and Evaluating Voice User Interfaces For The Home. In *Conference on Human Factors in Computing Systems*, 2013.

[176] Stephan Schlögl, Pierrick Milhorat, Gérard Chollet, and Jérôme Boudy. Designing Language Technology Applications: A Wizard of Oz Driven Prototyping Framework. In *EACL Conference of the European Chapter of the Association for Computer Linguistics*, pages 85–88, 2014.

[177] Marc Schröder and Jürgen Trouvain. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*, 6(4):365–377, 2003.

[178] Richard Schwartz and Scott Miller. Language understanding using hidden understanding models. In *International Conference on Spoken Language*, pages 997–1000, 1996.

[179] Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. GALAXY-II: a reference architecture for conversational system development. *International Conference on Spoken Language*, 1998.

[180] Vidura Seneviratne and Steve Young. The hidden vector state language model. *Unknown journal*, 2005.

[181] Abhinav Sethy, Panayiotis Georgiou, and Shrikanth Narayanan. Building topic specific language models from webdata using competitive models. *InterSpeech*, 2005.

[182] Candace L Sidner. An artificial discourse language for collaborative negotiation. In *Innovative Applications of Artificial Intelligence Conferences*, pages 814–819, 1994.

[183] Shweta Sinha, Shyam Sunder Agrawal, and Aruna Jain. Advances in voice enabled human machine communication. *International Journal of Computer Applications*, 2012.

[184] Mike A. Spaans. *On developing acoustic models using HTK*. PhD thesis, Delft University of Technology, 2004.

[185] Amanda J. Stent and Srinivas Bangalore. Statistical shared plan-based dialog management. In *International Conference on Spoken Language Processing*, 2008.

[186] Andreas Stolcke. SRILM-An extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, 2002.

[187] Andreas Stolcke and Elizabeth Shriberg. Statistical language modeling for speech disfluencies. In *International Conference on Acoustics, Speech and Signal Processing*, 1996.

[188] David Suendermann and Roberto Pieraccini. One year of contender: what have we learned about assessing and tuning industrial spoken dialog systems? *Workshop on Future directions and needs in the Spoken Dialog Community*, 2012.

[189] Stephen Sutton, Ronald Cole, Jacques De Villiers, Johan Schalkwyk, Pieter Vermeulen, Mike Macon, Yonghong Yan, Ed Kaiser, Brian Rundle, Khaldoun Shobaki, Paul Hosom, Alex Kain, Johan Wouters, Dominic Massaro, and Michael Cohen. Universal Speech Tools: The CSLU Toolkit. In *International Conference on Spoken Language Processing*, volume 98, pages 3221 – 3224, 1998.

[190] Blaise Thomson, Filip Jurčíček, Milica Gašić, Simon Keizer, François Mairesse, Kai Yu, and Steve Young. Parameter learning for POMDP spoken dialogue system. *Spoken Language Technology Workshop*, pages 259–264, 2010.

[191] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010.

[192] Maria Inés Torres. Stochastic Bi-Languages to model Dialogs. In *International Conference on Finite State Methods and Natural Language Processing*, pages 9–17, 2013.

[193] Maria Inés Torres, José-Miguel Benedí, Raquel Justo, and Fabrizio Ghigi. Modeling spoken dialog systems under the interactive pattern recognition framework. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 519–528. Springer Berlin Heidelberg, 2012.

[194] John Treichler. Signal Processing: A View of the Future, Part 1. *IEEE Signal Processing Magazine*, 26(2):116–118, 2009.

[195] John Treichler. Signal Processing: A View of the Future, Part 2. *IEEE Signal Processing Magazine*, 26(3):83–86, 2009.

[196] Gokhan Tur and Renato De Mori. *Spoken language understanding: Systems for extracting semantic information from speech.* John Wiley & Sons, 2011.

[197] Viswanath Venkatesh and Hillol Bala. Technology Acceptance Model 3 and a Research Agenda on Interventions. *Decision Sciences*, 39(2):273–315, 2008.

[198] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. *Sphinx-4: A flexible open source framework for speech recognition*. Sun Microsystems, 2004.

[199] Wayne Ward. Modelling non-verbal sounds for speech recognition. In *Workshop on Speech and Natural Language*, pages 47–50, 1989.

[200] Wayne Ward. Understanding spontaneous speech. In *Workshop on Speech and Natural Language*, pages 137–141, 1989.

[201] Wayne Ward. The CMU air travel information service: Understanding spontaneous speech. *DARPA Speech and Natural Language Workshop*, pages 127–129, 1990.

[202] Wayne Ward. Understanding spontaneous speech: the Phoenix system. In *International Conference on Acoustics, Speech and Signal Processing*, pages 127–129, 1990.

[203] Wayne Ward and Sunil Issar. Recent improvements in the CMU spoken language understanding system. In *Workshop on Human Language Technology*, 1994.

[204] Joseph Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[205] Graham Wilcock. WikiTalk: A spoken Wikipedia-based open-domain knowledge access system. In *International Conference on Computational Linguistics*, pages 57–70, 2012.

[206] Jason D. Williams, Pascal Poupart, and Steve Young. Factored partially observable Markov decision processes for dialogue management. *International Joint Conference on Artificial Intelligence*, 2005.

[207] Jason D. Williams, Pascal Poupart, and Steve Young. Partially observable Markov decision processes with continuous observations for dialogue management. *Recent Trends in Discourse and Dialogue*, pages 191–217, 2008.

[208] Jason D. Williams and Steve Young. Scaling up POMDPs for dialog management: the "summary POMDP" method. *Automatic Speech Recognition and Understanding Workshop*, 2005.

[209] Jason D. Williams and Steve Young. Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI). *Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, 2006.

[210] Jason D. Williams and Steve Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[211] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[212] Wayne Wobcke, Van Ho, Anh Nguyen, and Alfred Krzywicki. A BDI Agent Architecture for Dialogue Modelling and Coordination in a Smart Personal Assistant. In *International Conference on Intelligent Agent Technology*, pages 323–329, 2005.

[213] Wayne Wobcke, Anh Nguyen, Van Ho, and Alfred Krzywicki. The Smart Personal Assistant: An Overview. In *AAAI Spring Symposium: Interaction Challenges for Intelligent Assistants*, pages 135–136, 2007.

[214] Anna Wong, Anh Nguyen, and Wayne Wobcke. Robustness of a Spoken Dialogue Interface for a Personal Assistant. *International Conference on Intelligent Agent Technology*, pages 123–127, 2007.

[215] Xuchen Yao, Emma Tosch, Grace Chen, Elnaz Nouri, Ron Arstein, Anton Leuski, Kenji Sagae, and David Traum. Creating conversational characters using question generation tools. *Dialogue and Discourse*, 3(2):125–146, 2012.

[216] Steve Young. Probabilistic Methods in Spoken–Dialogue Systems. *Philosophical Transactions of the Royal Society of London*, 2000.

[217] Steve Young. Talking to machines (statistically speaking). *International Conference on Spoken Language Processing*, 2002.

[218] Steve Young. Using POMDPs for Dialog Management. In *Workshop on Spoken Language Technology*, 2006.

[219] Steve Young, Gunnar Evermann, Mark J. F. Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil C. Woodland. The HTK Book (for HTK Version 3.4), 2006.

[220] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

[221] Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. POMDP-based Statistical Spoken Dialog Systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

[222] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

[223] Victor Zue, Stephanie Seneff, James R. Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington. JUPlTER: a telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96, 2000.

Appendix A

Articles

HANDS-FREE SPEECH-SOUND INTERACTIONS AT HOME

P. Milhorat[1], D.Istrate[3], J. Boudy[2], G. Chollet[1]

[1]Télécom ParisTech, 37-39 rue Dareau, 75014, Paris, France
[2]Télécom SudParis, 9 rue Charles Fourier, 91011 Evry Cedex, France
[3]ESIGETEL, 1 Rue du Port de Valvins, 77210 Avon Cedex, France

ABSTRACT

This paper describes a hands-free speech/sound recognition system developed and evaluated in the framework of the CompanionAble European Project. The system is intended to work continuously on a distant wireless microphone and detect not only vocal commands but also everyday life sounds. The proposed architecture and the description of each module are outlined. In order to have good recognition and rejection rates, some constraints were defined for the user and the vocabulary was limited. First results are presented; currently project trials are underway.

Index Terms— speech recognition, sound processing, sound recognition, domotics.

1. INTRODUCTION

The CompanionAble European project aims at combining smart home functionalities with mobile robot abilities for dependent people. The robot is the front-end of the domotic system (turning on/off the lights, shutting/opening the curtains, playing/stopping music, etc) as well as an everyday helper. Supported by external sensors in the house (infra red sensors, door opening detectors, etc) and internal data (camera, sonar, etc), it's an assistant reacting to predefined scenarios (homecoming, video call, etc) or defined by the user himself (task reminder, pill dispenser, etc).

To achieve such variety of tasks, the device is equipped with a touch screen. A mobile tablet and a static screen on the kitchen wall are also available. These are the three means to access the common graphical user interface of the system.

Esigetel and the Mines-Télécom institute gave the robot its vocal interaction ability. A list of domotic commands have been extracted from practical experiments with end users.

Other applications, for instance the agenda, the cognitive training or the robot control are also accessed via vocal commands. In both cases, commands are not only words but full natural language sentences.

Lots of projects were about speech recognition; current commercial systems show us how the vocal interaction may be widely available in a near future. However, our work tries to solve the issues related to the distance to the microphone. In our configuration, we use a single microphone on top of the robot which can drive anywhere in the one-floor house. The noise environment is also unrestricted and traditional. Noise subtraction methods with dedicated microphone recording hypothetical noise sources are difficult to be applied to this real time changing environment.

The CompanionAble project is further detailed in the second part of this paper. Sections 3 and 4 are about the sound processing and classification process, then, in section 5, the speech recognition system is described. Section 6 presents the first evaluations. Conclusions and perspectives drawn from this work are presented in the final part.

2. COMPANIONABLE

CompanionAble stands for Integrated Cognitive Assistive & Domotic Companion Robotic Systems for Ability & Security. This project is funded by the European commission and is composed by 18 academic and industrial partners. Partners are from France, Germany, Spain, Austria, Belgium, the Netherlands and the United-Kingdom. The main objectives are:

- To combine mobile companion robot ability with smart home functionalities
- To support social connection for dependent people
- To improve the quality of life and the autonomy of elderly people

Esigetel and the Mines-Telecom institute are leaders, each, to develop a vocal interaction and a multimodal distress situation detector. They take part in the person localization within the house as well. This paper focuses on the acoustic work.

Currently, the project is tested by end users in SmH Eindhoven (Netherlands) and LabinHam in Gits (Belgium). They are invited to try the whole system for several consecutive days.

3. SOUND PROCESSING ARCHITECTURE

The sound is acquired continuously through two parallel systems: a first one which is able to detect and classify sound events between existing sound classes; another one

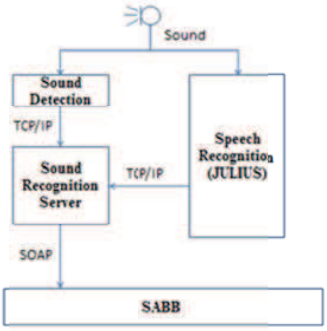


Figure 1 - Sound Processing Architecture

which is the speech recognition system. Figure 1 shows the communication between the sound and speech modules which is achieved using the TCP/IP protocol. The sound classification results and the vocal commands are sent to the CompanionAble architecture using SOAP protocol. The speech recognition output is filtered by the sound recognition system in order to avoid false alarms.

4. SOUND RECOGNITION

The sound recognition system is a two steps system: a detection module based on wavelet transform and a hierarchical recognition system (sound/speech and sound classification) based on GMM [2].

The sound classes used in CompanionAble trials were trained using sounds recorded with the CMT microphone (microphone under development from AKG) [3] in the Smart Homes (SmH) experimentation house. Currently the system has 5 sound classes: object fall, door ring, keys, cough and claps. The sound classes were chosen in order to help the CompanionAble system to identify distress situations and person activities.

The output of speech recognition is filtered by sound recognition module in order to avoid sending a wrong speech output instead of sound type identification. Because the two modules (sound and speech) are working in parallel, synchronization is needed. The sound module records in a buffer the three last sound/speech decisions associated with a timestamp. Then the decision of sending or rejecting the speech outputs within the server is effective between pairs of matching timestamps.

Each module was initially evaluated on pre-recorded data. For the sound the results for signals without noise was about 80% of good recognition rate [2]. The speech/no speech classification were experienced in SmH with a high rate near 95% of good classification.

5. SPEECH RECOGNITION

End-user difficulties, even inability to interact with the system using menus and touch screens justify the need for vocal interactions. Heavy cognitive or mobility troubles could make the system obsolete if it wasn't for the distant speech recognition. Three issues had been identified:

- Speech recognition in noisy uncontrolled environment
- Distant speech recognition
- Always-on speech recognition

Labs for practical experiments are based in the Netherlands and in Belgium, thus the human-machine interaction language is Dutch for the project.

Julius, developed by the Kawahara lab of Tokyo, was selected as the most appropriate recognition engine for a state-of-the-art speech recognition system [1]. It is able to process large-vocabulary search in real time, through a two-pass algorithm. It needs to be fed with N-gram language models and Hidden Markov Models (HMM) as acoustic models. The Julius engine can process the same audio input with several instances based on different language models.

The HMM of phones were trained on the *Corpus Gesproken Nederlands* (CGN). It is made of 800 hours of recorded and transcribed speech containing nearly 9 million words; this is the largest corpus for contemporary Dutch. Files are single speaker and multiple speakers recording, prompted or spontaneous speech.

Given the conditions (always on, distance to the microphone, uncontrolled noise environment) we present next some propositions to improve the robustness.

5.1. Trigger word

The dialog manager offered to lower the false positive rate with a trigger word. This word, when detected in the audio stream, increases the attention level which then decreases steadily according to time. When this level is positive, it triggers the recognition results analysis. For instance, the level is initially null: the speech recognition engine always processes the audio stream and outputs texts, as long as the trigger word can't be extracted from those segmented texts, transcriptions are rejected/ignored by the dialog manager. As soon as the attention level is above zero, the actual dialog starts and the manager analyses the received transcribed commands. While the dialog is sustained between the user and the system – either by repeating the trigger word or by evolution of the dialog – the attention level increases while silences, from the dialog manager point of view, decrease it. It can then reach its floor value and stop the input analysis. The selection of the attention word is important for the stability and liability of such a mechanism.

5.2. Speech/sounds classification

A speech recognition engine such as Julius search for the closest sequence of words matching the input audio observations given the probabilities contained in the acoustic model and in the language model. One may add a garbage model which will be the default match for unknown observation sequences or sub-sequences. In this application, every input is matched with a sequence of words. Thus sounds are processed as they were speech and a word sequence is returned. The sound classification prevents this to happen by discarding speech recognition results that occur while the stream has been classified as sound. It's a real-time process in parallel of the speech recognition one.

5.3. Acoustic adaptation

Acoustic adaptation methods have been studied at the earliest stage of the project [4]. Two adaptation methods were compared, namely: Maximum A Posteriori (MAP) and Maximum Likelihood Linear Regression (MLLR).
A language model was trained on a corpus of 57500 sentences derived from practical experiments and paraphrasing. The speaker is the same for the whole study, she has been previously recorded and the audio files are played through a loudspeaker. As expected, as only 10 phonetically balanced sentences are used, MLLR adaptation is the most suited technique. Without adaptation, 60% of the Julius' transcriptions are correct while this rate reach 70% with MAP adaptation and 73% with MLLR adaptation. Users go through MLLR adaptation before they use the system.

5.4. Language model combination

The first version of the speech recognition module was based on a single N-gram model trained on a 57658-sentence corpus. The acoustic model was adapted to fit the voice characteristics of the users using the MLLR method. This first system presented too much false positives, i.e. unwanted commands, when put to practical tests.
In order to improve both the recognition and rejection rates, a filter, described next, was implemented.
The dialog is based on frames [5]. These frames contain sub-dialogue graphs and transitions between states are triggered by the robot internal state/variables and the user inputs (vocal commands, buttons or/and sensors). A frame is enabled when at least one of its activation conditions is fulfilled; these are the same kinds of variables than the intra-frames ones. Thus one can build a dialogue hierarchy: the root frame which is initially enabled contains all the activation events to enable the sub frames and terminal states allow the sub frames to hand over the control to the main frame.
The sub frames have been clustered within eight classes. Each class lists all the vocal commands which are allowed and can be interpreted in the compound frames. A language model is build from those lists.
A 9^{th} language model is trained on the activation commands and is associated to the main frame.
Even while the speech recognition module doesn't receive information about the state of the dialogue, nine instances of the recognition engine run at the same time and deliver transcriptions of the input audio stream.
This language model selection process improves the good recognition rates for the application commands but on the other hand doesn't solve the rejection issues for out-of-application sentences.

5.5. Similarity test

Similarity between two recognizers' hypothesis is an extended Levenshtein distance. This is the total number of operation (substitution, deletion, insertion) to transform a sentence in another one. Furthermore it is normalized with the count of word in the sentences. Depending of the relative value of this distance, given a threshold, the hypothesis recognized by an engine fed with a specific language model is accepted or discarded. This test is used to:

- Confirm good recognition: a well recognized command according to both the general decoder and the specific decoder is validated. The exact specific decoder's hypothesis is sent
- Reject wrong hypothesis: a command recognized only by the general decoder is rejected.
- Correct partially correct hypothesis: a command recognized by a specific decoder while the general decoder outputs a close match is corrected: the specific hypothesis is sent

The general language model must, in this setup, recognize the sequences of word contained in the specific language models. One needs to add the whole set of commands in the training corpus of such a general model. We introduced a weight for these additional sentences which has been experimentally defined to be 1000: the commands were added 1000 times.
Finally, the test is not effective between one hypothesis for each decoder. We found out that it is better to use the n-best ones; it improves the good recognition rate:

- One hypothesis is outputted for specific decoder because of the size of their language model
- Several hypothesis (3 in our application) are produced by the general decoder and then fed to the similarity test

All this improvements were implemented. A first evaluation of those is presented next.

6. EVALUATIONS

A test corpus has been recorded in SmH with 5 speakers.

Each one of them spoke 58 sentences: 10 adaptation sentences, 20 application commands, 22 out-of-application sentences and 6 modified sentences. The modification is actually a deletion of one or several words in a command from the application.

The experiment setup is shown on Figure 2, Audio Dutch sequences are played through a loudspeaker and recorded by a microphone. The top loudspeaker was used for the experiment second phase and simulated noisy conditions. The sound level was of about 60 dBA which is the level of an average speaker standing about one meter away from the listener.

The first phase was intended to set the value of the commands 'weight in the general model and the number of hypothesis from the general recognition process for the similarity measure.

For vocal commands allowed by the application, the baseline, which is fed with a language model trained on all the commands, recognizes 15% of them.

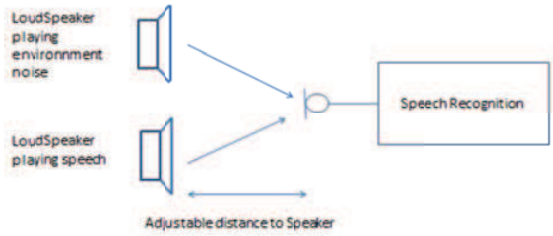


Figure 2 – Experimental setup for the tests

The similarity test addition improved these results up to 20% but one could notice an increase of the false-positive rate. The best system can recognizes commands with 85% of accuracy and never gives false-positive cases.

Every out-of-application sentences have been rejected by the system.

Modified commands are most of the time validated.

System	Recognition rate	False-positives rate
Baseline + adaptation	15	0
Baseline + adaptation + similarity test (commands' weight: 1; general decoder hypothesis: 1)	20	10
Baseline + adaptation + similarity test (commands' weight: 1000; general decoder hypothesis: 1)	55	0
Baseline + adaptation + similarity test (commands' weight: 1000; general decoder hypothesis: 3)	85	0

Table 1. Recognition rate and false-positive rate for in-application commands

System	Recognition rate	False-positives rate
Baseline + adaptation	15	0
Baseline + adaptation + similarity test (commands' weight: 1; general decoder hypothesis: 1)	20	10
Baseline + adaptation + similarity test (commands' weight: 1000; general decoder hypothesis: 1)	55	0
Baseline + adaptation + similarity test (commands' weight: 1000; general decoder hypothesis: 3)	85	0

Table 2. Recognition rate and false-positive rate for out-of-application commands

System	Recognition rate	False-positives rate
Baseline + adaptation	15	0
Baseline + adaptation + similarity test (commands' weight: 1; general decoder hypothesis: 1)	20	10
Baseline + adaptation + similarity test (commands' weight: 1000; general decoder hypothesis: 1)	55	0
Baseline + adaptation + similarity test (commands' weight: 1000; general decoder hypothesis: 3)	85	0

Table 3. Recognition rate and false-positive rate for mixed commands

In a second phase of the experimental process, the noise robustness was tested. The top loudspeaker plays ambient sounds.

As expected, the performances dropped. The good recognition rate is lowered as well as the rejection rate.

System	Recognition rate	False-positives rate
Wash machine	74	11
Dutch speaker	53	11
Music	47	5
Crowd	42	11

Table 4. Recognition rate and false-positive rate for in-application commands

System	Recognition rate	False-positives rate
Wash machine	0	0
Dutch speaker	0	0

Music	0	0
Crowd	0	3.64

Table 5. Recognition rate and false-positive rate for out-of-application commands

System	Recognition rate	False-positives rate
Wash machine	40	0
Dutch speaker	60	0
Music	20	0
Crowd	60	0

Table 6. Recognition rate and false-positive rate for mixed commands

7. CONCLUSIONS AND PESPECTIVES

The set up presented on this paper aims at providing spoken input for a companion robot within a smart home environment. As the robot is always on, so is the speech recognizer. Given these constraints, the most important characteristic to keep in mind is the robustness of the recognition. This combines both a good recognition rate but also accurate rejection criteria.

A trade-off between these two aspects has to be found. Is it acceptable to erroneously recognize a command? Can the user be asked to repeat utterances? During trials, it has been noticed that false positives could mean trouble and disturb the user. To solve this issue, the command/sentence set to be recognized has been restricted, this yields two other problems. Intended users are elderly and dependant people who get some trouble remembering specific commands. Furthermore, they could get quickly upset if the robot doesn't recognize their orders and think that this is useless, ignoring this functionality.

We proposed to experiment a combination of language models to improve the system accuracy.

A new general language model has been built from a read Dutch subset of the CGN corpus. Let's assume that it is able to recognize any Dutch utterances. Then another pass works on a restricted specific model with close vocabulary. The similarity between both resulting sentences, computed as a variation of the Levenshtein distance, behaves as a filter for acceptation/rejection.

A closer collaboration with the dialog manager would also bring more ways of refinement and filtering. The dialog manager of the CompanionAble project implemented in the companion robot follows a finite state automaton clustered in frames. Except for the root/main state which activates sub-frames and so is always active, we can select a specific language model built from acceptable sentences given a frame. Thus 10 restricted models have been created, one for each frame and one for the "main" frame. The dialog manager listens to the recognition process outputs, filtering them with what the current state(s) allow(s).

This more elaborated system proved to be robust enough to allow a good recognition rate as well as limited false positive cases. However, informal experiments showed its weakness when it comes to reject short commands, i.e. one-word sentences. The use of the robot's attention with the trigger word prevents this to happen.

8. ACKNOWLEDGMENTS

This work is supported by the FP7 European CompanionAble Project. We thank AKG in Vienna and SmartHome in Eindhoven for their assistance. We also thank Daniel Caon and Pierre Sendorek for their help in the first implementation of the speech recognition system.

9. REFERENCES

[1] A. Lee, "The Julius Book", http://julius.sourceforge.jp/juliusbook/en/, 2008

[2] J.E. Rougui, D. Istrate, W. Souidene, "Audio Sound Event Identification for distress situations and context awareness", EMBC2009, September 2-6, Minneapolis, USA, 2009, pp. 3501-3504

[3] J.E. Rougui, D. Istrate, W. Souidene, M. Opitz et M. Riemann, "Audio based surveillance for cognitive assistance using a CMT microphone within socially assistive technology", EMBC2009, September 2-6, Minneapolis, USA, 2009, pp.2547-2550

[4] D. Caon, T. Simonnet, J. Boudy and G. Chollet, "vAssist: The Virtual Interactive assistant for Daily Home-care", pHealth conference, 8th International Conference on Wearable Nano and Macro Technologies for Personalized Health. June 29th-Jult 1st 2011.Lyon, France

[5] S. Müller, C. Schroeter, H.-M. Gross, "Aspects of user specific dialog adaptation", International Scientific Colloquium, Ilmenau, Germany, 2010.

Multi-step Natural Language Understanding

Pierrick Milhorat, Stephan Schlögl, Gérard Chollet
Institut Mines-Télécom
Télécom ParisTech, CNRS LTCI
Paris, France
{lastname}@enst.fr

Jérôme Boudy
Institut Mines-Télécom
Télécom SudParis
Paris, France
boudy@telecom-sudparis.eu

Abstract

While natural language as an interaction modality is increasingly being accepted by users, remaining technological challenges still hinder its widespread employment. Tools that better support the design, development and improvement of these types of applications are required. This demo presents a prototyping framework for Spoken Dialog System (SDS) design which combines existing language technology components for Automatic Speech Recognition (ASR), Dialog Management (DM), and Text-to-Speech Synthesis (TTS) with a multi-step component for Natural Language Understanding (NLU).

1 Introduction

Recently speech and other types of natural language are experiencing an increased acceptance when being used for interacting with 'intelligent' computing systems. This trend is particularly reflected by products such as Apple's *Siri*[1], Google's *Now*[2] and Nuance's *Dragon Solutions*[3]. While these applications demonstrate the industry's vision of how we should be interacting with our current and future devices, they also highlight some of the great challenges that still exist. One of these challenges may be seen in the fact that Automatic Speech Recognition (ASR) remains a highly error-prone technology which influences subsequent natural language processing components such as Natural Language Understanding (NLU) and Dialog Management (DM) and leads to often unsatisfying user experiences. Hence we require appropriate tools that better support the testing and studying of language as an interaction modality and consequently allow us to build better, more user-centered applications.

This demo presents our approach of developing a prototyping tool for Spoken Dialog Systems (SDS). Our solution is particularly focusing on the natural language understanding aspect of SDS design. The overall framework is composed of a set of existing open-source technology components (i.e. ASR, DM, TTS) which are expanded by several additional NLP modules responsible for natural language understanding as well as generation. The following sections first provide a general overview of the entire framework and then focus particularly on the NLU part of our solution and the different sub-modules it integrates.

2 Spoken Dialog System Design

A state-of-the-art SDS usually consists of a set of technology components that are integrated to form a consecutive processing chain. Starting on the input side the ASR module produces a hypothesis about the orthographic content of a spoken utterance. The NLU takes this recognized utterance and converts it into a machine readable command or input Dialog Act (DA). The DM processes this input DA and sends the relevant output DA to the Natural Language Generation (NLG) component. The NLG is then responsible for converting the output DA into appropriate natural language text. Finally, the Text-to-Speech (TTS) synthesis component takes the text transmitted by the NLG and speaks it to a user.

According to this general architecture different open-source language components have been integrated to form a loosely coupled SDS framework. The framework includes ASR performed by the Julius Large Vocabulary Continuous Speech Recognition engine[4], dialog management based on the Disco DM library (Rich, 2009; Rich

[1] http://www.apple.com/ios/siri/
[2] http://www.google.com/landing/now/
[3] http://www.nuance.com/dragon/
[4] http://julius.sourceforge.jp/en_index.php

Proceedings of the SIGDIAL 2013 Conference, pages 157–159,
Metz, France, 22-24 August 2013.

and Sidner, 2012) and TTS achieved through the MARY Text-to-Speech Synthesis Platform[5]. Additionally, we have integrated the WebWOZ Wizard of Oz Prototyping Platform[6] (Schlögl et al., 2010) in order to allow for the simulation of (flawless) natural language understanding. Expanding these existing components we have then developed as a set of modules responsible for actual system-based natural language processing. The following section describes these modules in more detail and highlights the types of challenges they try to overcome.

3 Natural Language Understanding

Within the processing chain of a spoken/text-based dialog system, the NLU component is the link between the wide and informal communication space of a user's input and the formal and rather restrictive semantic space that can be processed by the DM (Mori et al., 2007). Trying to bridge these two spaces we have connected several modules to form an NLU processing segment whose different modules are described below.

3.1 Semantic Parsing

First we use a Semantic Parsing (SP) module to convert the transcribed speech provided by the ASR into so-called Semantic Frames (SFs). To achieve this mapping Jurčíček et al. (2009) designed a Transformation-Based Learning Semantic Parser (Brill, 1995) which we adapted to integrate it with our framework. The algorithm applies an ordered set of rules to hypothetical [*utterance, SF*] pairs in order to find the closest matching SF.

3.2 Semantic Unification

Next we use what we call the Semantic Unifier and Reference Resolver (SURR) module to convert input SFs into SFs that can be processed by the DM input interface. To do this we implemented a bottom-up search algorithm for rewriting trees whose nodes contain lists of valued slots. The algorithm looks for a group of root nodes that can be reached in the forest (i.e. the existing number of trees) by transforming an input SF's set of slots according to the given rewriting rules. It succeeds when all slots can be rewritten into a root list of slots. This module is supported by external knowledge sources such as for example the context in which an utterance has been produced (i.e. it receives input from the Context Catcher module described below). Furthermore it could call operating system functions, sensor readings [7] or other knowledge sources capable of providing relevant data, in order to resolve and disambiguate input. For instance, special-valued slots like 'date=today' are dynamically resolved to the correct data type and value, making the NLU more sensitive to its surrounding environment.

3.3 Context Inclusion

In order to optimize information exchange Human-Human interactions usually build up a common knowledge between dialog participants. This inherent grounding process can be compared to the dialog history recorded in an SDS's DM. Using these recordings we have introduced a so-called Context Catcher (CC) module. The way this module is currently working is as follows: The DM requests information from the user to progress through the task-oriented dialog. The user replies without specifying the type of data he/she is providing, the overall intent of the utterance or the relation to any dialog slot. The CC evaluates the request expressed by the DM and consequently updates various parameters of the SURR component. Consequently the SURR is able to provide a better, more context-specific mapping between raw SFs provided by the SP module and the expected slots to be filled by the DM component.

3.4 Dialog Act Conversion

An SDS's DM expects formal meaning representations to be converted to actual dialog moves or Dialog Acts (DA); similar to parametrized dialog commands. A DA is the smallest unit of deterministic action to support the dialogue flow. The number of DAs that are available at any given point is finite, dynamic and depends on the current state of the dialog (Note: Here a state does not refer to a 'real' state, such as the ones used in Markov Decision Processes or Partially Observable Markov Decision Processes, but rather to a general status of the dialog). In other words, two input utterances carrying the same meaning may lead to different consequences depending on a given dialog state. The right action, i.e. the accurate DA, is to be determined by the NLU component. As there

[5] http://mary.dfki.de/

[6] https://github.com/stephanschloegl/WebWOZ

[7] Note: At the moment sensor readings are not implemented as they are currently not available in the developing environment

is usually a many-to-many matching between SFs and actual DAs we integrated an additional Dialog Act Converter (DAC) module. This module uses the context to generate a list of expected slots for which a user may provide a value (i.e. it converts possible DAs to SFs). Then a matching between the actual inputs and the expectations is applied in order to find the most probable DA.

4 Supporting Mixed Initiatives

SDS dialog designs usually run along an initiative scale that ranges from user-driven to strictly machine-driven interaction. In the case of a machine-driven dialog a user has to follow the requests of the system. Interactions that lie out of the scope of this dialog design are not understood and may either be discarded or, in the worst case, lead to a system failure. Despite this potential for failure, machine-driven designs make the dialog easier to control and thus less prone to errors, yet, due to the lack of adaptability exposed by the system, also less human-like. On the other hand, pure user-driven dialog designs minimize the functional range of a system as they only react to commands without assuring their functional integrity.

The above described modular approach to NLU aims to support a mixed initiative design where a system's integrity and its goals are sufficiently defined; the user, however, is not restricted by the type and amount of spoken input he/she can use to interact. To offer this type of interaction the system needs to handle three kinds of potential mis-usages: (1) out-of-application cases, (2) out-of-dialog cases and (3) out-of-turn cases. To address the first one our training corpus has been augmented so that it includes examples of garbage SFs. As a result an out-of-application utterance triggers a generic reply from the system, notifying the user that he/she is outside the scope of the application. In the case where a user stays within the scope of the application but tries to initiate a new unrelated dialog (i.e. out-of-dialog case), the DM's stack of tasks is incremented with the new dialog. The system will lead the user back to the previous topic once the newly added one is completed. Finally, as for the out-of-turn cases i.e. the cases where a user would answer a system request with a non-expected utterance such as an over-complete one, the NLU process, retrieving the DM's expectations, discards unrelated or over-complete information.

5 Demo Description

Focusing on the NLU aspect of the SDS pipeline this demo will demonstrate how the different modules described above (i.e. SP, SURR, CC, and DAC) work together. An application scenario from the ambient assisted living domain (i.e. the operation of a 'Pillbox' application) will serve as an example use case. It will be shown how the natural language input potentially recognized by an ASR component is further interpreted by our NLU processing segment. All the steps discussed in Section 3 will be visible.

6 Conclusion

In this paper we described a set of NLU components that were integrated as part of a loosely coupled SDS. Separate modules for semantic parsing, semantic unification and reference resolution, context inclusion as well as dialog act conversion have been described. Furthermore we have highlighted how our system offers support for mixed-initiative dialog interactions. A first test of this NLU processing chain showed that the use of our multi-component approach is feasible, and we believe that this solution can be seen as a valuable test and development framework for natural language processing research.

References

E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*.

F. Jurčíček, F. Mairesse, M. Gašić, S. Keizer, B. Thomson, K. Yu, and S. Young. 2009. Transformation-based Learning for semantic parsing. *Proceedings of INTERSPEECH*, pages 2719–2722.

R. De Mori, F. Béchet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur. 2007. Spoken language understanding: A survey. *Proceedings of ASRU*.

C. Rich and C. L. Sidner. 2012. Using collaborative discourse theory to partially automate dialogue tree authoring. *Intelligent Virtual Agents*, pages 327–340.

C. Rich. 2009. Building task-based user interfaces with ANSI/CEA-2018. *Computer*.

S. Schlögl, G. Doherty, S. Luz, and N. Karamanis. 2010. WebWOZ: A Wizard of Oz Prototyping Framework. In *Proceedings of ACM EICS*, pages 109–114.

What If Everyone Could Do It? A Framework for Easier Spoken Dialog System Design

Pierrick Milhorat
Institut Mines-Télécom
Télécom ParisTech
CNRS LTCI
Paris, France
milhorat@telecom-paristech.fr

Stephan Schlögl
Institut Mines-Télécom
Télécom ParisTech
CNRS LTCI
Paris, France
schlogl@telecom-paristech.fr

Jérôm Boudy
Institut Mines-Télécom
Télécom SudParis
Paris, France
jerome.boudy@telecom-sudparis.eu

Gérard Chollet
Institut Mines-Télécom
Télécom ParisTech
CNRS LTCI
Paris, France
chollet@telecom-paristech.fr

ABSTRACT

While Graphical User Interfaces (GUI) still represent the most common way of operating modern computing technology, Spoken Dialog Systems (SDS) have the potential to offer a more natural and intuitive mode of interaction. Even though some may say that existing speech recognition is neither reliable nor practical, the success of recent product releases such as Apple's *Siri* or Nuance's *Dragon Drive* suggests that language-based interaction is increasingly gaining acceptance. Yet, unlike applications for building GUIs, tools and frameworks that support the design, construction and maintenance of dialog systems are rare. A particular challenge of SDS design is the often complex integration of technologies. Systems usually consist of several components (e.g. speech recognition, language understanding, output generation, etc.), all of which require expertise to deploy them in a given application domain. This paper presents work in progress that aims at supporting this integration process. We propose a framework of components and describe how it may be used to prototype and gradually implement a spoken dialog system without requiring extensive domain expertise.

Author Keywords

SDS Design; Language Technology Components; WOZ.

ACM Classification Keywords

H.5.2 User Interfaces: Natural language; H.5.2 User Interfaces: Prototyping; D.5.2 User Interfaces: Voice I/O

General Terms

Human Factors; Design.

EICS'13, June 24–27, 2013, London, United Kingdom.

INTRODUCTION

Spoken Dialog Systems (SDS) are booming, with products such as Apple's *Siri*[1], Google's *Voice Search*[2] or Nuance's *Dragon Solutions*[3] demonstrating how current (and future) technologies may change the way we interact with our devices. Even though a lot of these potential applications might also be achievable using traditional Graphical User Interfaces (GUI), a reasonably 'intelligent' computer system that (sufficiently) understands spoken input would simply convey a better user experience [23]. Yet, the design of this type of systems is complex and so we see a pressing demand for tools and techniques that better support this task. SDSs usually consist of several language technology components, ranging from speech recognition and generation to dialog management and artificial intelligence. Building functioning solutions may therefore require sufficient expertise in several different domains. While some tool-support for standalone components exists (e.g. [14, 27, 30, 8, 9, 26, 4]) only few attempts have been undertaken to generate a more holistic framework for SDS design (e.g. [15, 16, 7, 3]).

This paper discusses an SDS prototyping framework that has been implemented by our research group. The goal of our approach is to exclude 'hand-crafting' work as much as possible and use a combination of machine learning algorithms and Wizard of Oz (WOZ) experimentation [13] to build SDS solutions from scratch. Integrating existing open-source technology components with WOZ we aim for the creation of a flexible and easy to use prototyping environment, that can be used not only by speech engineers, but also by designers/researchers outside the signal processing community. The paper starts with an overview of the proposed framework architecture which is followed by a description of its different components. After that we discuss our current employment of the framework and conclude the paper with planned future directions.

[1] http://www.apple.com/ios/siri/
[2] http://www.google.com/mobile/voice-search/
[3] http://www.nuance.com/dragon/

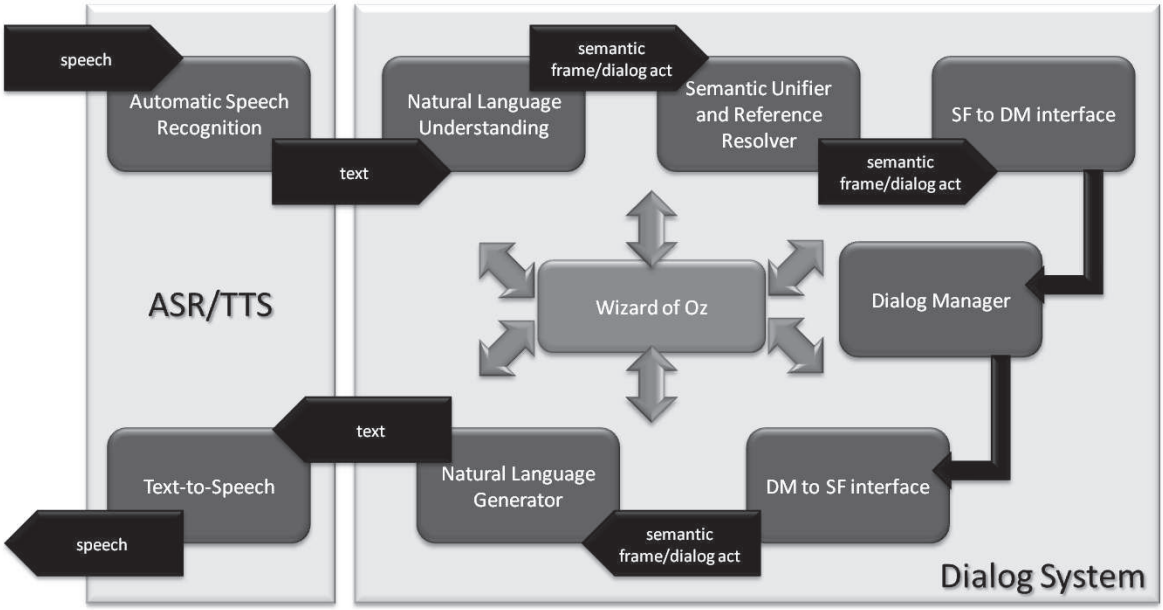


Figure 1. A framework architecture for Spoken Dialog System design

FRAMEWORK COMPONENTS

The overall architecture of our prototyping framework closely resembles the state of the art processing chain of a modern SDS (cf. Figure 1). On the input side we find the Automatic Speech Recognition (ASR) module, a Natural Language Understanding (NLU) component, as well as a novel component which we call the Semantic Unifier and Reference Resolver (SURR). The output side consists of a Natural Language Generation (NLG) component and a Text-to-Speech synthesis (TTS) module. The core of the system is represented by a Dialog Manager (DM) which is connected to the input and output chain via two formatting interfaces. These interfaces offer additional flexibility with respect to possible future framework extensions (e.g. a potential integration of multi-modality).

At runtime the ASR module decodes speech input, producing natural language text utterances and passes them on to the NLU component. The NLU component then extracts from these utterances so-called Semantic Frames (SF), which consist of a goal and *0 to n* instantiated slots. Next, the SURR component filters these SFs, replaces relative values like dates, times or locations by absolute ones and resolves references. Then an interface component translates the SURR output into a format that can be processed by the DM. The DM, which represents the core of the overall system, is responsible for keeping track of the dialog progress, taking into account the given context, and consequently triggers the request for additional input; i.e. it is aware of the current tasks and therefore demands the relevant variables to be defined. It takes the output of the SURR and, based on the currently loaded task model, selects appropriate actions (i.e. it initiates utterances to be produced by the NLG component or commands to be sent to a back-end application). Again, a dedicated formatting interface is used to translate the DM output into a format that can be interpreted by the NLG component. Such consequently produces the requested text utterance. Finally, the TTS module takes the NLG output and converts it into synthesized speech.

In order to offer this work flow we have integrated a set of open-source language technology components, augmented by various 'home-made' software modules, into a flexible SDS prototyping framework. The following sections will describe the different components of this framework and their roles in some more detail, and highlight which extensions and adaptations were necessary in order to create a cohesive interaction pipeline.

Automatic Speech Recognition Module

SDSs are different from other dialog systems in that speech represents their single interaction modality. Thus, an SDS's first processing stage has to generate hypotheses about the orthographic content that is encoded in a user's spoken input. Despite decades of research and commercial deployment this processing is still regarded as highly error-prone. Current best practice is to search for the best matching sequence of stochastic models using the digitized input signal. Mel-Frequency Cepstrum Coefficients (MFCC) (and their deltas) are widely used descriptors for such speech signal analyses (e.g. [6, 2, 1]). The distribution of the coefficients' vectors for the contextualized phonemes or triphones (i.e. the smallest units of the processed sound signal) are usually encoded as Hidden Markov Models (HMM) [11], which were trained from already transcribed speech segments. These models constitute the first ingredient for building a working ASR module – the so called Acoustic Model (AM). Next, in order to construct words out of a sequence of phonemes, a Pronouncing Dictionary (PD) is required, which consists of the decomposition of a language's words into phonemic

units. Finally, the last ingredient that is necessary to build the ASR module is a so-called Language Model (LM) which provides probabilities for given word sequences to appear in a sentence. Those probabilities are based on existing linguistic structures and encoded as n-grams. The combination of the three knowledge sources (i.e. AM, PD and LM) is then used by the recognition engine to produce one or several hypothesis of recognized text for a given (segmented) speech signal [1].

Given these requirements one may argue that building ASR systems for distinct application scenarios is time consuming and very much dependent on both the availability of required knowledge sources (i.e. AM, PD and LM), and the quality and amount of data that was used to construct them. Yet, existing Large Vocabulary Continuous Speech Recognition systems (LVCSR) often already cover a great amount of general purpose vocabulary (as long as their training has been performed on such data). Hence, extending such a general system (and its knowledge sources) to fit the vocabulary space of a specific application scenario may be quicker and more effective than building an entirely new recognizer from scratch. What is needed, however, are appropriate interfaces that allow for the adaptation of the general models so that they better facilitate the recognition of expected utterances related to a specific application scenario. Milhorat et al. [17] proposed a filtering method to favor such a recognition of 'correct' utterances while discarding mis-recognized or out-of-context ones. Results could then further be augmented with features like the dialog state, the dialog history and, a user's personalized settings, and eventually be used to dynamically update/replace an LVCSR's general engine configuration with a more specific, application dependent one.

In order to offer a solution that allows for such a dynamic adaptation of knowledge sources our prototyping framework integrates the Julius ASR engine, an LVCSR engine developed by the Kawahara Lab at Kyoto University [14]. The current setup supports the recognition of spoken input in English, French, Spanish and Dutch. In addition we have acquired the necessary databases to build recognizers for German and Italian. Using this setting we plan to create adapted language models for a number of application scenarios, including the speech-based operation of a calendar program, the use of communication services such as email and text messages, and the interaction with several health and well-being applications (e.g. a well-being diary).

Natural Language Understanding Component

Although all uni-modal dialog systems work with only one input modality (i.e. either direct text input or text recognized by an ASR component) the meaning representation they employ can differ greatly between solutions [9, 18]. The outptut that has to be produced by an integrated NLU component therefore depends on the purpose of the overall system as well as its DM formalism. Specific implementations can take on various forms and notations. For our prototyping framework we have chosen a frame-based semantic representation of language understanding. Semantic Frames (SF) are often used because of their versatility. An SF (cf. Figure 2) consists of a goal (i.e. the user's intent) and is further defined by a number of relevant parameters, represented by slot-value pairs. Given a textual input, the task of and SF-based NLU component is to select a matching SF (i.e. a goal and its parameters) from a predefined set of possibilities. It does this by applying a number of rules which are usually learned from an annotated corpus.

Figure 2. The example of a Semantic Frame (SF)

The NLU component we have integrated employs an algorithm developed by Jurcicek et al. [12]. It is based on sequential transformation rules which are applied to find a match between an input utterance and an SF. Rules consist of triggers and transformation operations. A trigger contains one or more conditions such as an n-gram or a skipping bigram in the user utterance, a goal value, or a slot-type in the (temporary) paired SF. The transformation is applied if all the conditions of a rule's trigger match the input utterance-SF pair. An utterance to be processed is initialized with the default dialog act i.e. no slot and the most common goal as determined by the annotated training corpus. The training algorithm then looks for one rule that maximizes the value of the optimization function (i.e. it follows a transformation-based learning principle). In our case the optimization measure is the distance between each temporary SF and the 'true' SF in the corpus. This is computed as the sum of required addition, deletion and substitution operations (i.e. Levenshtein distance). Once this best rule is found, it is applied to the current state of the corpus and the algorithm is re-initiated for the resulting new training database. The process is stopped when the best rule's increase of the optimization function is below a given threshold.

Semantic Unifier and Reference Resolver

The Semantic Unifier and Reference Resolver (SURR) is not a standard SDS component but rather one of the features that was needed to fill the gap between the NLU component integrated with our prototyping framework and its DM component. In particular, it transforms the NLU output, which is out of context, so that it can be processed by the following DM. For example, if we want to add a valid event entry to a calendar application the system usually requires an event name (i.e. a title), a starting as well as an ending point in time (i.e. a date and a time) and maybe an optional note. A user, however, might interact with the system as follows:

- User: "Add the birthday of my daughter, on Saturday the 15th of November from 2 pm"
- System: *[asks the user for the ending-point-in-time slot's value]* "When will it be finished?"
- User: "I think I'll be there for 6 hours"

In this situation, even if the system would create an SF with a duration slot of 6 hours instead of an ending time, the DM would not be able to process the data as it requires a precise ending. What we see here is an SF space difference between the semantic interpreter (NLU) and the decision-making component (DM). To solve this mismatch we would need to augment the entire dialog task model, which consequently might also require significant changes to be made to the back-end application. Instead, however, our framework uses a dynamic mapping component (i.e. the SURR) that allows for a duration slot to be converted into an ending-point-in-time slot. We call this process the semantic unification. Furthermore we use what we call a reference resolution process to convert the 'tomorrow' that is used in the above example into 'today's date incremented by one day'. Both operations, semantic unification and reference resolution, are contained in the same tree structures which are searched by the SURR algorithm. These trees are handcrafted from situations that happen in experiments with real users, and then further expanded according to a designer's/researcher's ideas. For instance, after having implemented the 'tomorrow' branch one may think of adding the 'yesterday' one.

The current version of the SURR module is based on data collected through a set of initial experiments. It employs a tree-climbing algorithm that is applied to a structure of additive and converting branches. Every link between nodes represents a predicate. Figure 3 shows an example of a tree and Figure 4 its associated implementation. The initial function looks for *1 to n* slot-value pair(s) for which a transforming predicate exists and subsequently applies the defined operation. The resulting (transformed) SF is then processed again and such is repeated until no further predicate match is found. The algorithm succeeds if the final SF contains only those pairs that are declared as roots. All parameters of an SF which cannot be replaced by a root slot (i.e. where the algorithm fails) are subsequently discarded.

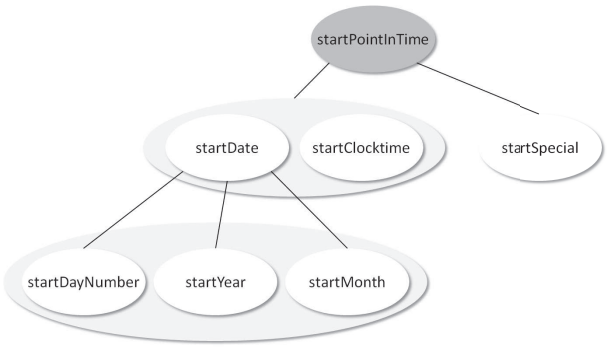


Figure 3. The tree structure of the Semantic Unifier and Reference Resolver (SURR)

Dialog Manager

To date several probabilistic DM components are available (e.g. [10, 28]). Yet, most of them require a significant amount of data to produce viable results, and their scalability is often limited to a few slots, user dialog acts and system actions. An alternative can be found in fully deterministic DM components whose functional breadth is pre-defined. Such,

```
/**root definitions*/
root([slot(startPointInTime, _)]).

/**start point in time(current date + current clocktime) -> start special(now)*/
rewrite([slot(startPointInTime, C)], [slot(startSpecial, [now])]) :-
        append(A, B, C), append(D, E, A), append(F, G, E),
        slot(currentMonth, D), slot(currentDayNumber, F),
        slot(currentYear, G), slot(currentClocktime, B).

/**start point in time -> start date + start clocktime*/
rewrite([slot(startPointInTime, P)], [slot(startDate, D), slot(startClocktime, C)]) :-
        append(D, C, P).

/**start date -> start day number + start month + start year*/
rewrite([slot(startDate, D)], [slot(startMonth, M), slot(startDayNumber, Dn), slot(startYear, Y)]) :-
        append(X, Y, D), append(M, Dn, X).
```

Figure 4. An example implementation of the Semantic Unifier and Reference Resolver (SURR)

however, requires greater knowledge of the supported dialog space and is therefore only suitable for well defined interaction domains. Since the goal of our framework is to support the development of dialog systems for specific application scenarios we decided to integrate Disco [21, 22], a representative of the later approach. It requires a task model compliant with the ANSI CEA-2018[4] standard, which essentially demands a recursive decomposition of tasks into atomic actions. Disco integrates a so-called inference engine which, if provided with one or more task models, is able to manage a mixed-initiative dialog. It processes a hierarchy of tasks (applying plan recognition), guiding the user towards the completion of macro tasks (consisting of several subtasks). Planning is performed automatically, supported by static task models and a dynamic focus stack. Task models contain the task structure, the temporal constraints for the dialog and the data flow within the models. They are implemented in XML and usually require expertise to be built. In order to help with their creation we investigated two notation languages. These languages aim at the automatic extraction of suitable task models based on the description of the given back-end application, where the back-end applications is represented by a form-filling service with attached commands. The first such language is a set of first-order logic formulas. It enables the designer/researcher to specify incompatibilities between slots, as well as optional and mandatory slot attributes. While such certainly helps the design process its application is somewhat limited. Additional manual edits are still required in order to support all the essential information a task model might need to encode. Hence a second, more advanced language is currently under development, which supports conditional relationships between slots, reusable sub-application descriptions, and computed values. Here, an application's command is described as a form containing an ID, a set of slots, sub-forms (i.e. links to other forms) and an action triggered by the completion of the form. Sub-form attributes are boolean optional, ignore and default, which respectively set the linked form to non mandatory, ignored (in the case the applicable condition is not fulfilled), or default (in the case of ambiguity). This process may allow for a richer formalism and should enable the designer/researcher to focus more thoroughly on the actual application.

[4] http://www.ce.org/Standards/Standard-Listings/R7-Home-Network-Committee/CEA-2018-(ANSI).aspx

Formatting Interfaces (SF to DM and DM to SF)

In addition to the earlier highlighted semantic ambiguities which exist between NLU and DM (we tackle them with the described SURR component), we often also find certain formatting incompatibilities between those two components (as well as between the DM and the following NLG component). Such is usually caused by the use of different input/output interface standards or diverging forms of knowledge representation. Generally the task of a DM component is to trigger output-dialog-acts and accompanying actions based on input provided by the NLU. The anticipated input as well as the produced output are, however, context dependent so that the current dialog state is often required for better disambiguation. To tackle this problem we have introduced two formatting interfaces; one of which translates an SF (delivered by the SURR) into a context-specific input-dialog-act (i.e. factoring in the current dialog state), and a second one that takes the output-dialog-act delivered by the DM and translates it back into an SF (i.e. the format that can be processed by our NLG component). While these interfaces do not modify the actual input/output content they can be regarded as necessary formatting components, implemented as an overlay to the actual DM. As such, they also offer more flexibility with respect to the modularization of our framework (Note: A future replacement of single components might require additional input/ output formatting).

Natural Language Generation Component

While NLG is generally an important aspect of an SDS it is currently not our main area of interest. Our framework therefore only implements a very basic generation engine. It uses the output of the DM (i.e. the output that has been converted by the output formatting interface described above) to select a human-readable response sentence form a set of possible templates. Each template uses a goal ID that is matched with the dialog act produced by the DM. The SDS designer has to provide at least as many templates (cf. Fig. 5) as dialog acts exist. In case there are more possible templates for a given dialog act, the NLG component randomly selects one and forwards it to the TTS.

```
goal:text /slot1/ text /slot2/ text.
time:Today is the /dayNuber/th of /month/.
```

Figure 5. Natural Language Generator templates

Text-to-Speech Synthesis Module

Finally, in order to generate speech from the text fragments produced by the NLG component, our framework integrates the OpenMary TTS [19, 25]; a state-of-the-art, open source, synthesis platform which supports several languages. We currently use the platform to produce speech output in German, Italian, French and English.

Wizard of Oz Component

One last important aspect of our proposed framework architecture is the integration of a Wizard of Oz (WOZ) component. WOZ constitutes a prototyping method that uses a human operator, the so-called wizard, to simulate a system (or part of it) in order to collect relevant interaction data [5]. To support this task we have integrated the WebWOZ prototyping platform [24]; a tool that permits the wizard to replace one or several components of an SDS. Such should offer an easy and efficient solution for various sorts of data gathering. For example, the training corpus for our NLU component consists of possible inputs and its matching outputs. Replacing this component by a human wizard who transforms spoken input into relevant dialogue acts (i.e. SFs), may alleviate the fastidious work of manually searching and annotating corpus data that matches a given application domain.

CURRENT FRAMEWORK EMPLOYMENT

The framework described above is currently used to build a multi-lingual SDS for an application scenario situated in the ambient assisted living domain. Experiments are conducted in which the WOZ component acts as a substitution for the ASR as well as the NLU component. Doing this we are able to collect various types of interaction data (mainly training data that is used for building and improving the NLU component and user experience data that helps to obtain initial end-user feedback). While our initial sessions are in French, experiments in German and Italian are planned for the next couple of month. Once sufficient data for a language is collected, one only needs to re-configure the ASR and re-train the NLU to integrate it with the system. Such demonstrates the flexibility we are aiming for with our framework composition. Another aspect of this flexibility is reflected by the amount of control the human operator (i.e. the wizard) can take over. Set-ups in which *1-n* parts of the framework are simulated/augmented/controlled should allow for accurate refinements of faulty or weak components as well as support user studies at any stage of the development process; an aspect which, we believe, may enable also non-experts to use our framework as a means for designing and building novel SDS solutions.

CONCLUSION AND FUTURE WORK

We presented a flexible SDS prototyping framework that aims to support the easy and quick construction of voice user interfaces for different application scenarios. The implementation of this framework is achieved through the integration of a set of interchangeable open-source language technology components. While the different components are not by default ready to be used with any application domain, their configuration and adaptation to fit a specific purpose requires only little knowledge and expertise.

Future work will focus on the adaptability and flexibility of the presented framework, particularly exploring its employment by non-expert users. Furthermore we will investigate possible ways of improving single framework components. For example, we aim for increasing the robustness of the ASR by using the feedback produced by post-processing components (i.e. NLU, SURR, DM). Another planned improvement is the use of parametric HMMs [29, 20]. Those can be controlled by a set of external shared parameters and therefore would match more closely the acoustic phenomenons of spoken language. Finally, we are also investigating the use of several speech recognition hypothesis.

ACKNOWLEDGEMENTS

The research presented in this paper is jointly supported by vAssist, a project funded by the European Ambient Assisted Living Joint Programme and the National Funding Agencies from Austria, France and Italy (AAL-2010-3-106), and ARHOME, a French national research project.

REFERENCES

1. Anusuya, M. A., and Katti, S. K. Front end analysis of speech recognition: a review. *International Journal of Speech Technology 14*, 2 (2011), 99–145.
2. Baker, J. M., Deng, L., Glass, J., Khudanpur, S., Lee, C.-h., Morgan, N., and O'Shaughnessy, D. Research Developments and Directions in Speech Recognition and Understanding, Part 1. *IEEE Signal Processing Magazine 26*, 3 (2009), 75–80.
3. Bohus, D., Raux, A., Harris, T. K., Eskenazi, M., and Rudnicky, A. I. Olympus: an open-source framework for conversational spoken language interface research. In *Proc. of ACL-HLT* (2007).
4. Churcher, G. E., Atwell, E. S., and Souter, C. Dialogue management systems: a survey and overview. *Research Report Series - University of Leeds School of Computer Studies* (February 1997).
5. Dahlbäck, N., Jönsson, A., and Ahrenberg, L. Wizard of oz studies - why and how. In *Proc. of ACM IUI* (1993), 193–200.
6. Davis, S. B., and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics Speech and Signal Processing 28*, 4 (1980), 357–366.
7. Galibert, O., Illouz, G., and Rosset, S. Ritel: an open-domain, human-computer dialog system. In *Proc. of INTERSPEECH* (2005), 909–912.
8. He, Y., and Young, S. Semantic processing using the Hidden Vector State model. *Computer Speech & Language 19*, 1 (2005), 85–106.
9. He, Y., and Young, S. Spoken language understanding using the hidden vector state model. *Speech Communication 48*, 3-4 (2006), 262–275.
10. Henderson, J., and Lemon, O. Mixture model POMDPs for efficient handling of uncertainty in dialogue management. In *Proc. of ACL-HLT* (2008), 73–76.
11. Juang, B.-H., and Rabiner, L. R. Hidden Markov models for speech recognition. *Technometrics 33*, 3 (1991), 251–272.
12. Jurčíček, F., Mairesse, F., Gašić, M., Keizer, S., Thomson, B., Yu, K., and Young, S. Transformation-based Learning for semantic parsing. *Evaluation* (2009), 2719–2722.
13. Kelley, J. F. An empirical methodology for writing User-Friendly Natural Language computer applications. In *Proc. of ACM CHI* (1983), 193–196.
14. Lee, C., Jung, S., and Lee, G. G. Robust dialog management with n-best hypotheses using dialog examples and agenda. In *Proc. of ACL-HLT* (2008), 630–637.
15. Leuski, A., Pair, J., and Traum, D. How to talk to a hologram. In *Proc. of IUI* (2006), 360–362.
16. Leuski, A., Patel, R., Traum, D., and Kennedy, B. Building effective question answering characters. *Proc. of SIGDIAL* (2009), 18–27.
17. Milhorat, P., Istrate, D., Boudy, J., and Chollet, G. Hands-free speech-sound interactions at home. In *Proc. of EUSIPCO* (2012), 1678–1682.
18. Mori, R. D., Béchet, F., Hakkani-Tur, D., McTear, M., Riccardi, G., and Tur, G. Spoken language understanding: A survey. In *Proc. of IEEE ASRU* (2007).
19. Pammi, S., Charfuelan, M., and Schröder, M. Multilingual voice creation toolkit for the MARY TTS platform. In *Proc. of LREC* (2010).
20. Radenen, M., and Artieres, T. Contextual hidden markov models. In *Proc. of ICASSP* (2012), 2113–2116.
21. Rich, C. Building task-based user interfaces with ANSI/CEA-2018. *Computer 42*, 8 (2009), 20–27.
22. Rich, C., and Sidner, C. L. Using collaborative discourse theory to partially automate dialogue tree authoring. In *Proc. of IVA* (2012), 327–340.
23. Schalkwyk, J., Beeferman, D., Beaufays, F., Byrne, B., Chelba, C., Cohen, M., Garret, M., and Strope, B. Google search by voice : A case study. *Visions of Speech: Exploring New Voice Apps in Mobile Environments, Call Centers and Clinics 2* (2010), 1–35.
24. Schlögl, S., Doherty, G., Karamanis, N., and Luz, S. WebWOZ: a wizard of oz prototyping framework. In *Proc. of ACM EICS* (2010), 109–114.
25. Schröder, M., and Trouvain, J. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology* (2003).
26. Seneviratne, V., and Young, S. The hidden vector state language model. In *Proc. of INTERSPEECH* (2005), 1–4.
27. Stolcke, A. SRILM-An extensible language modeling toolkit. In *Proc. of ICSLP* (2002).
28. Williams, J. D., and Young, S. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language 21*, 2 (2007), 393–422.
29. Wilson, A. D., and Bobick, A. F. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 9 (1999), 884–900.
30. Young, S., Evermann, G., Gales, M. J. F., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. C. The HTK Book (for HTK Version 3.4), 2006.

Designing Language Technology Applications: A Wizard of Oz Driven Prototyping Framework

S. Schlögl
MCI Management Center Innsbruck
Management, Communication & IT
Innsbruck, AUSTRIA
schlogl@mci.edu

P. Milhorat*, G. Chollet*, J. Boudy†
Institut Mines-Télécom
*Télécom ParisTech & †Télécom SudParis
Paris, FRANCE
milhorat@telecom-paristech.fr

Abstract

Wizard of Oz (WOZ) prototyping employs a human wizard to simulate anticipated functions of a future system. In Natural Language Processing this method is usually used to obtain early feedback on dialogue designs, to collect language corpora, or to explore interaction strategies. Yet, existing tools often require complex client-server configurations and setup routines, or suffer from compatibility problems with different platforms. Integrated solutions, which may also be used by designers and researchers without technical background, are missing. In this paper we present a framework for multi-lingual dialog research, which combines speech recognition and synthesis with WOZ. All components are open source and adaptable to different application scenarios.

1 Introduction

In recent years Language Technologies (LT) such as Automatic Speech Recognition (ASR), Machine Translation (MT) and Text-to-Speech Synthesis (TTS) have found their way into an increasing number of products and services. Technological advances in the field have created new possibilities, and ubiquitous access to modern technology (i.e. smartphones, tablet computers, etc.) has inspired novel solutions in multiple application areas. Still, the technology at hand is not perfect and typically substantial engineering effort (gathering of corpora, training, tuning) is needed before prototypes involving such technologies can deliver a user experience robust enough to allow for potential applications to be evaluated with real users. For graphical interfaces, well-known prototyping methods like sketching and wire-framing allow for obtaining early impressions and initial user feedback. These low-fidelity prototyping techniques do not, however, work well with speech and natural language. The Wizard of Oz (WOZ) method can be employed to address this shortcoming. By using a human 'wizard' to mimic the functionality of a system, either completely or in part, WOZ supports the evaluation of potential user experiences and interaction strategies without the need for building a fully functional product first (Gould et al., 1983). It furthermore supports the collection of domain specific language corpora and the easy exploration of varying dialog designs (Wirén et al., 2007). WOZ tools, however, are often application dependent and built for very specific experimental setups. Rarely, are they re-used or adapted to other application scenarios. Also, when used in combination with existing technology components such as ASR or TTS, they usually require complex software installations and server-client configurations. Thus, we see a need for an easy 'out-of-the-box' type solution. A tool that does not require great technical experience and therefore may be used by researchers and designers outside the typical NLP research and development community. This demo is the result of our recent efforts aimed at building such an integrated prototyping tool.

We present a fully installed and configured server image that offers multi-lingual (i.e. English, German, French, Italian) ASR and TTS integrated with a web-based WOZ platform. All components are open-source (i.e. adaptable and extendable) and connected via a messaging server and a number of Java programs. When started the framework requires only one single script to be executed (i.e. there is a separate script for each language so that the components are started using the right parameters) in order to launch a WOZ driven system environment. With such a pre-configured setup we believe that also non-NLP experts are able to successfully conduct extended user studies for language technologies applications.

Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 85–88,
Gothenburg, Sweden, April 26-30 2014.

2 Existing Comparable Tools

Following the literature, existing tools and frameworks that support prototyping of language technology applications can be separated into two categories. The first category consists of so-called Dialogue Management (DM) tools, which focus on the evaluation of Language Technologies (LTs) and whose primary application lies in the areas of NLP and machine learning. Two well-known examples are the CSLU toolkit (Sutton et al., 1998) and the Olympus dialogue framework (Bohus et al., 2007). Others include the Jaspis dialogue management system (Turunen and Hakulinen, 2000) and the EPFL dialogue platform (Cenek et al., 2005). DM tools explore the language-based interaction between a human and a machine and aim at improving this dialogue. They usually provide an application development interface that integrates different LTs such as ASR and TTS, which is then used by an experimenter to specify a pre-defined dialogue flow. Once the dialogue is designed, it can be tested with human participants. The main focus of these tools lies on testing and improving the quality of the employed technology components and their interplay. Unlike DM tools, representatives from the second category, herein after referred to as WOZ tools, tend to rely entirely on human simulation. This makes them more interesting for early feedback, as they better support the aspects of low-fidelity prototyping. While these applications often offer more flexibility, they rarely integrate actual working LTs. Instead, a human mimics the functions of the machine, which allows for a less restrictive dialogue design and facilitates the testing of user experiences that are not yet supported by existing technologies. Most WOZ tools, however, should be categorized as throwaway applications i.e. they are built for one scenario and only rarely re-used in other settings. Two examples that allow for a more generic application are SUEDE (Klemmer et al., 2000) and Richard Breuer's WOZ tool[1].

While both DM and WOZ tools incorporate useful features, neither type provides a full range of support for low-fidelity prototyping of LT applications. DM tools lack the flexibility of exploring aspects that are currently not supported by technology, and pure WOZ applications often depend too much on the actions of the wizard, which can lead to unrealistic human-like behaviour and inconsistencies with its possible bias on evaluation results. A combination of both types of tools can outweigh their deficiencies and furthermore allow for supporting different stages of prototyping. That is, a wizard might complement existing technology on a continuum by first taking on the role of a 'controller' who simulates technology. Then, in a second stage one could act as a 'monitor' who approves technology output, before finally moving on to being a 'supervisor' who only overrides output in cases where it is needed (Dow et al., 2005). However, to allow for such variation an architecture is required that on the one hand supports a flexible use of technology components and on the other hand offers an interface for real-time human intervention.

[1] http://www.softdoc.de/woz/index.html

3 Integrated Prototyping Framework

In order to offer a flexible and easy to use prototyping framework for language technology applications we have integrated a number of existing technology components using an Apache ACTIVEMQ messaging server[2] and several Java programs. Our framework consists of the JULIUS Large Vocabulary Continuous Speech Recognition engine[3], an implementation of the GOOGLE SPEECH API[4], the WEBWOZ Wizard of Oz Prototyping Platform[5] and the MARY Text-to-Speech Synthesis Platform[6]. All components are fully installed and connected running on a VIRTUAL BOX server image[7] (i.e. Ubuntu 12.04 LTS Linux Server). Using this configuration we offer a platform that supports real-time speech recognition as well as speech synthesis in English, French, German and Italian. Natural Language Understanding (NLU), Dialog Management (DM), and Natural Language Generation (NLG) is currently performed by the human 'wizard'. Respective technology components may, however, be integrated in future versions of the framework. The following sections describe the different components in some more detail and elaborate on how they are connected.

[2] http://activemq.apache.org/
[3] http://julius.sourceforge.jp/en_index.php
[4] http://www.google.com/intl/en/chrome/demos/speech.html
[5] https://github.com/stephanschloegl/WebWOZ
[6] http://mary.dfki.de/
[7] https://www.virtualbox.org/

3.1 Automatic Speech Recognition

The JULIUS open-source Large Vocabulary Continuous Speech Recognition engine (LVCSR) uses n-grams and context-dependent Hidden Markov Models (HMM) to transform acoustic input into text output (Lee et al., 2008). Its recognition performance depends on the availability of language dependent resources i.e. acoustic models, language models, and language dictionaries. Our framework includes basic language resources for English, German, Italian and French. As those resources are still very limited we have also integrated online speech recognition for these four languages using the Google Speech API. This allows for conducting experiments with users while at the same time collecting the necessary data for augmenting and filling in JULIUS language resources.

3.2 Text-to-Speech Synthesis

MARY TTS is a state-of-the-art, open source speech synthesis platform supporting a variety of different languages and accents (Schröder and Trouvain, 2003). For the here presented multilingual prototyping framework we have installed synthesized voices for US English (cmu-slt-hsmm), Italian (istc-lucia-hsmm), German (dfki-pavoque-neutral) as well as French (enst-dennys-hsmm). Additional voices can be downloaded and added through the MARY component installer.

3.3 Wizard of Oz

WebWOZ is a web-based prototyping platform for WOZ experiments that allows for a flexible integration of existing LTs (Schlögl et al., 2010). It was implemented using modern web technologies (i.e. Java, HTML, CSS) and therefore runs in any current web browser. It usually uses web services to integrate a set of pre-configured LT components (i.e. ASR, MT, TTS). For the presented prototyping framework, however, we have integrated WebWOZ with our ASR solution (i.e. the combined Google/JULIUS engine) and MARY TTS. Consequently ASR output is displayed in the top area of the wizard interface. A wizard is then able to select an appropriate response from a set of previously defined utterances or use a free-text field to compose a response on the fly. In both cases the utterance is sent to the MARY TTS server and spoken out by the system.

3.4 Messaging Server and Gluing Programs

In order to achieve the above presented integration of ASR, WOZ and TTS we use an Apache ACTIVEMQ messaging server and a number of Java programs. One of these programs takes the output from our ASR component and inserts it into the WebWOZ input stream. In addition it publishes this output to a specific ASR ActiveMQ queue so that other components (e.g. potentially an NLU component) may also be able to process it. Once an ASR result is available within WebWOZ, it is up to the human wizard to respond. WebWOZ was slightly modified so that wizard responses are not only sent to the internal WebWOZ log, but also to a WIZARD ActiveMQ queue. A second Java program then takes the wizard responses from the WIZARD queue and pushes them to a separate MARY queue. While it may seem unnecessary to first take responses from one queue just to publish them to another queue, it allows for the easy integration of additional components. For example, we have also experimented with a distinct NLG component. Putting this component between the WIZARD and the MARY queue we were able to conduct experiments where a wizard instead of sending entire text utterance would rather send text-based semantic frames (i.e. a semantically unified representation of a user's input). Such shows the flexibility of using the described queue architecture. Finally we use a third Java program to take text published to the MARY queue (i.e. either directly coming from the wizard or produced by an NLG component as with one of our experimental settings) and send it to the MARY TTS server. Figure 1 illustrates the different framework components and how they are connected to each other.

4 Demo Setup

The optimal setup for the demo uses two computer stations, one for a wizard and one for a test user. The stations need to be connected via a LAN connection. The test user station runs the prototyping framework, which is a fully installed and configured Virtual Box software image (Note: any computer capable of running Virtual Box can serve as a test user station). The wizard station only requires a modern web browser to interact with the test user station. A big screen size (e.g. 17 inch) for the wizard is recommended as such eases his/her task. Both stations will be provided by the authors.

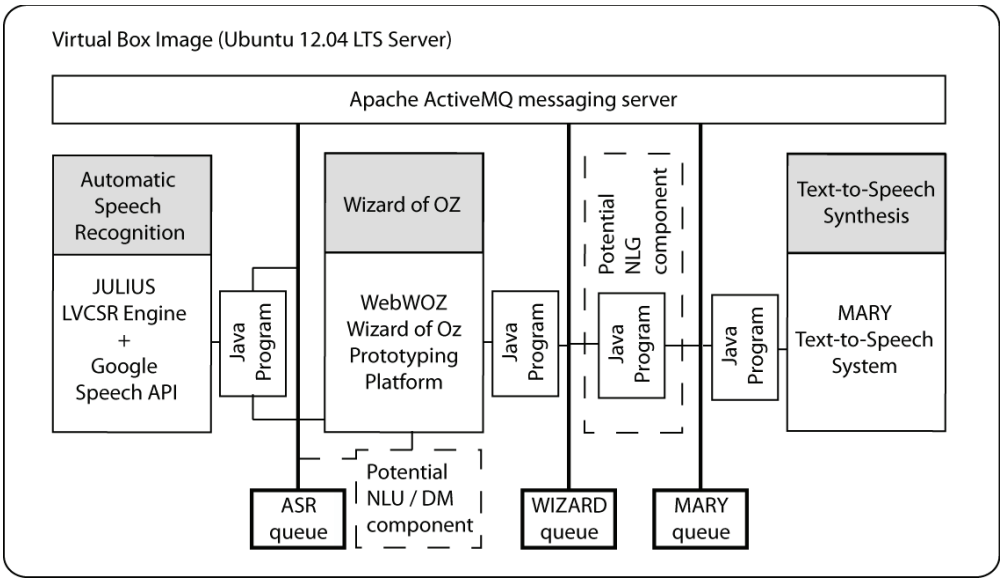


Figure 1: Prototyping Framework Components.

5 Summary and Future Work

This demo presents an integrated prototyping framework for running WOZ driven language technology application scenarios. Gluing together existing tools for ASR, WOZ and TTS we have created an easy to use environment for spoken dialog design and research. Future work will focus on adding additional language technology components (e.g. NLU, DM, NLG) and on improving the currently limited ASR language resources.

Acknowledgments

The presented research is conducted as part of the vAssist project (AAL-2010-3-106), which is partially funded by the European Ambient Assisted Living Joint Programme and the National Funding Agencies from Austria, France and Italy.

References

D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I. Rudnicky. 2007. Olympus: An open-source framework for conversational spoken language interface research. In *Proc. of NAACL-HLT*, pages 32–39.

P. Cenek, M. Melichar, and M. Rajman. 2005. A framework for rapid multimodal application design. In *Proceedings of TSD*, pages 393–403.

S. Dow, B. Macintyre, J. Lee, C. Oezbek, J. D. Bolter, and M. Gandy. 2005. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4(4):18–26.

J. D. Gould, J. Conti, and T. Hovanyecz. 1983. Composing letters with a simulated listening typewriter. *Communications of the ACM*, 26(4):295–308.

S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang. 2000. SUEDE: A wizard of oz prototyping tool for speech user interfaces. In *Proc. of UIST*, pages 1–10.

C. Lee, S. Jung, and G. G. Lee. 2008. Robust dialog management with n-best hypotheses using dialog examples and agenda. In *Proc. of ACL-HLT*, pages 630–637.

S. Schlögl, G. Doherty, N. Karamanis, and S Luz. 2010. WebWOZ: a wizard of oz prototyping framework. In *Proc. of the ACM EICS Symposium on Engineering Interactive Systems*, pages 109–114.

M. Schröder and J. Trouvain. 2003. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*.

S. Sutton, R. Cole, J. de Vielliers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, B. Rundle, K. Shobaki, P. Hosom, A. Kain, J. Wouters, D. Massaro, and M. Cohen. 1998. Universal speech tools: The CSLU toolkit.

M. Turunen and J. Hakulinen. 2000. Jaspis- a framework for multilingual adaptive speech applications. In *Proc. of ICSLP*, pages 719–722.

M. Wirén, R. Eklund, F. Engberg, and J. Westermark. 2007. Experiences of an In-Service Wizard-of-Oz Data Collection for the Deployment of a Call-Routing Application. In *Proc. of NAACL-HLT*, pages 56–63.

Appendix B

Defense slides

Institut Mines-Télécom

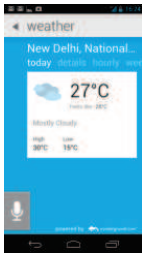
An Open-source Framework for Supporting the Design and Implementation of Natural-language Spoken Dialog Systems

Pierrick Milhorat

17 décembre 2014

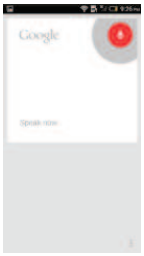
Recent trend in the vocal user interfaces

Echo
(Amazon)



Maluuba
(Maluuba)

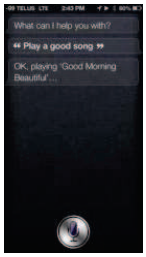
Cortana
(Microsoft)



Google Now
(Google)

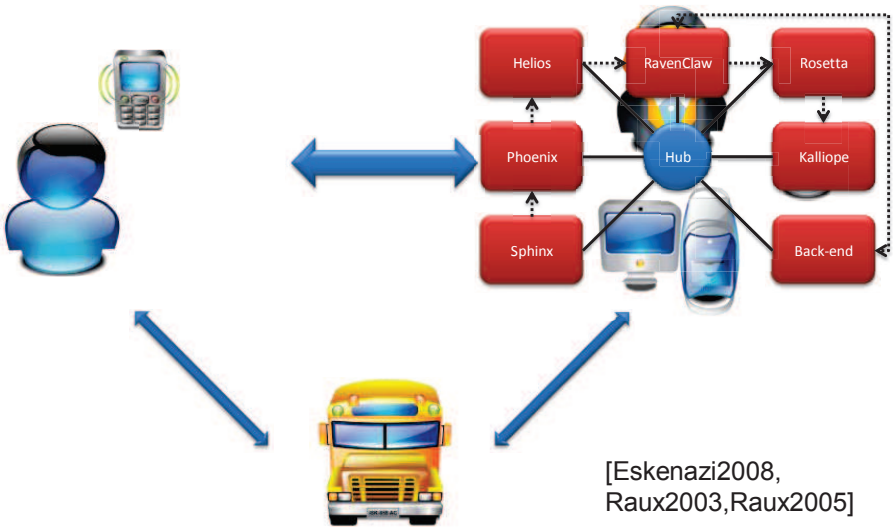
S Voice
(Samsung)

Voice Mate
(LG)



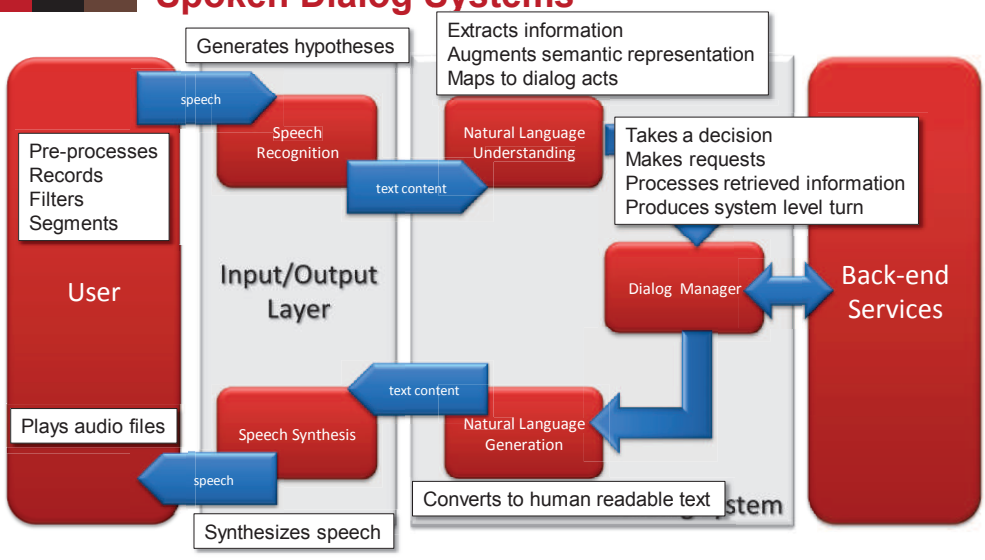
Siri
(Apple)

A stand-out instance of a deployed research system: Let's Go Public!



[Eskenazi2008, Raux2003,Raux2005]

Spoken Dialog Systems



Research questions

- How can we make the human-machine spoken interaction more reliable?
- What are the important features for a spoken dialog system to improve its human-like appearance?
- How can we support the development of spoken dialog systems?

Claims

- A modular open-source platform for spoken dialog systems
- A step towards continuous listening for spoken interaction
- A sub-system to map natural-language utterances to situated parameterized dialog acts
- The linked form-filling language: a new paradigm to create and update task-based dialog models

Projects

Start Date: December 2011
Duration: 36 months

Start Date: January 2008
Duration: 48 months

Fond Unique Interministériel N°12

Start Date: October 2011
Duration: 24 months

A modular open-source platform for spoken dialog systems

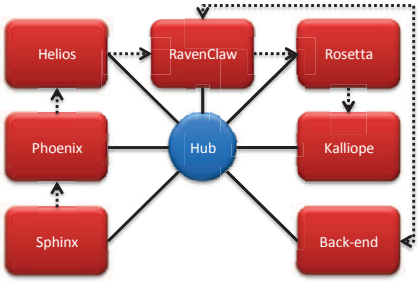
Commercial vs research solutions

Not well performing

Expertise requirement

Free

Open-source

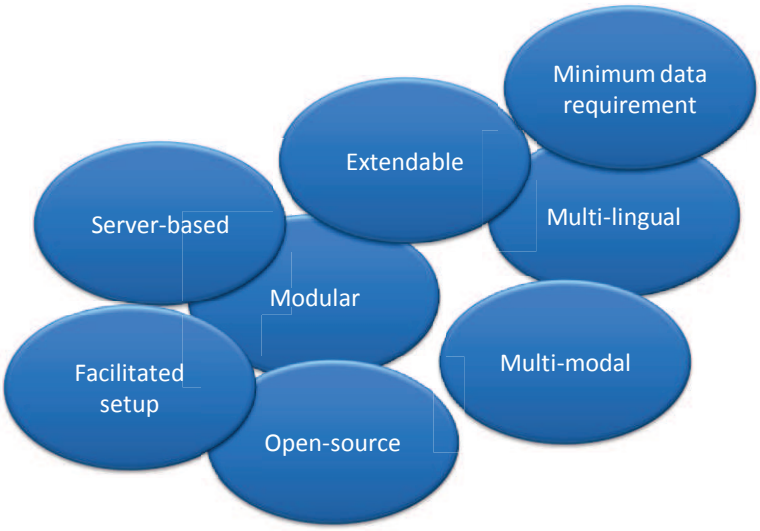


Black boxes

Costly

Fit the 'average' user

Desired characteristics



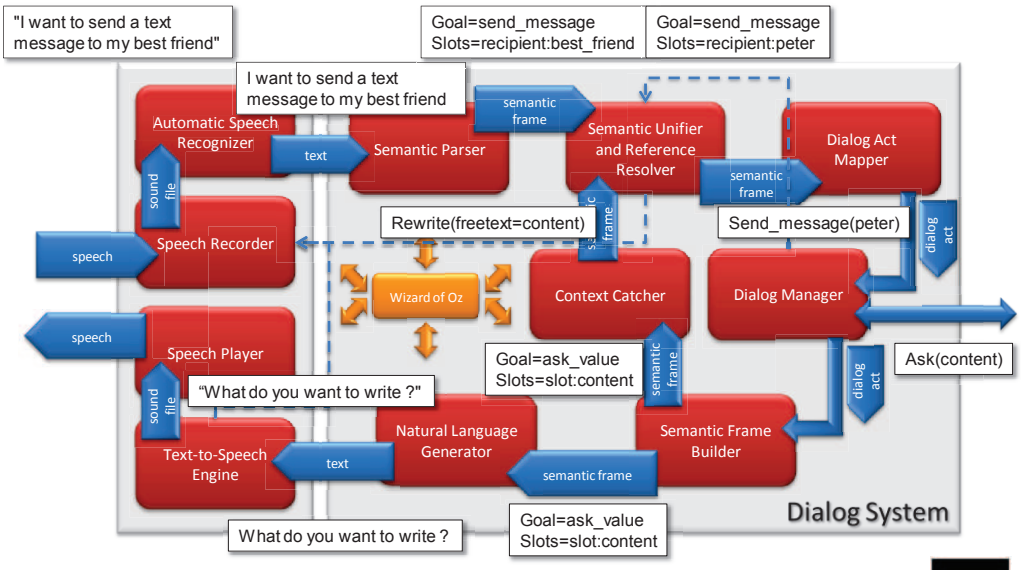
Compared to…

✓Has the characteristic
× Does not have the characteristic
- Information is unavailable

	Server-based	Modular	Extendable	Open-source	Minimum data requirement	Multi-lingual	Multi-modal	Facilitated setup
CLSU Toolkit [McTear1998]	×	✓	✓	✓	✓	✓	×	✓
GALAXY-II [Seneff1998]	✓	×	×	✓	✓	✓	×	×
Voice XML	✓	×	×	✓	✓	✓	×	✓
Olympus [Bohus2007]	✓	✓	✓	✓	✓	✓	×	×
OpenDial* [Lison2012]	✓	✓	✓	✓	✓	✓	×	✓
Siri-like systems	✓	-	×	×	-	✓	×	-

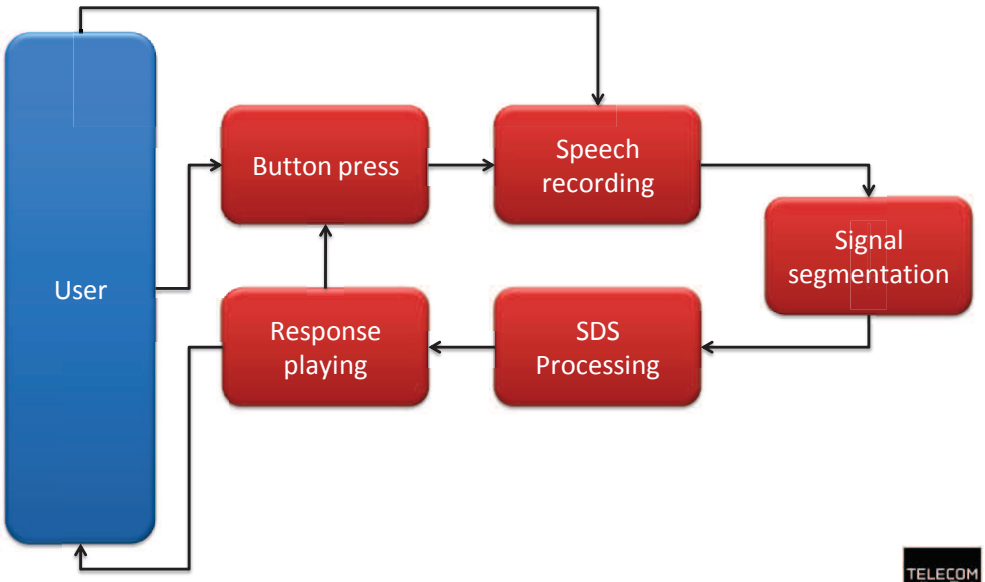
*v. 0.95, released on April 4, 2014

Architecture

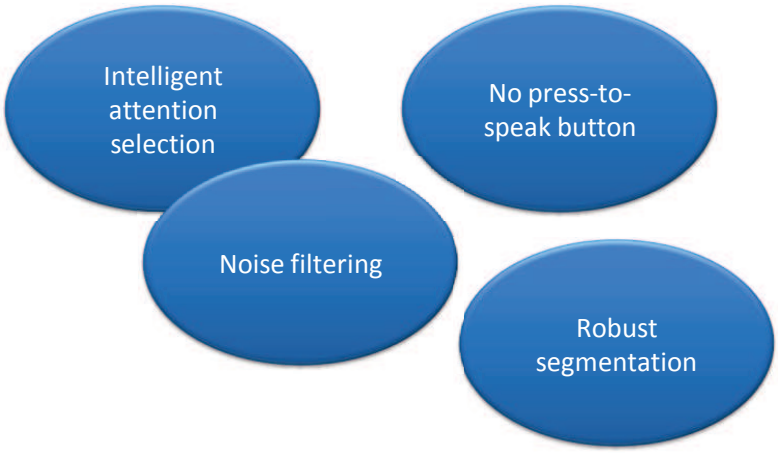


Towards continuous listening for spoken interaction

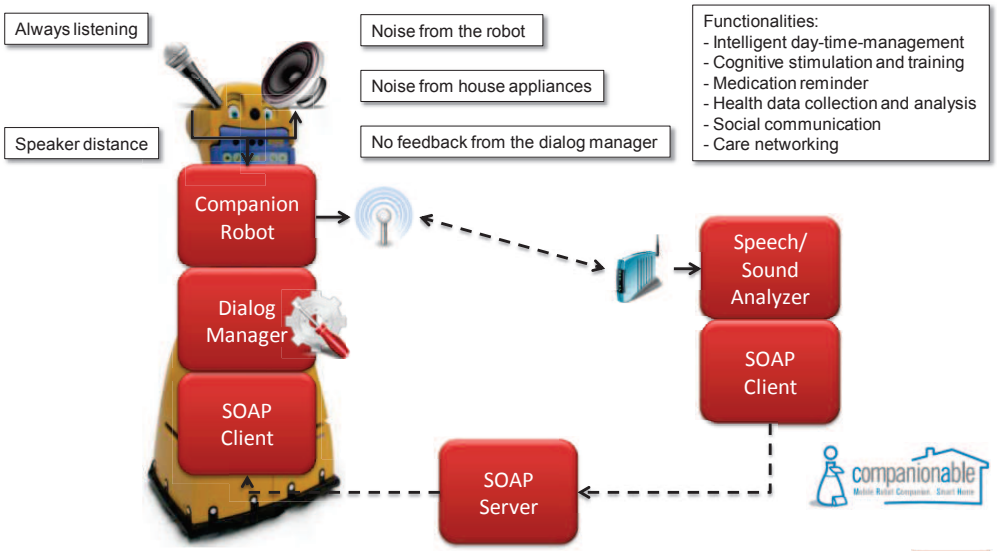
Most common listening method



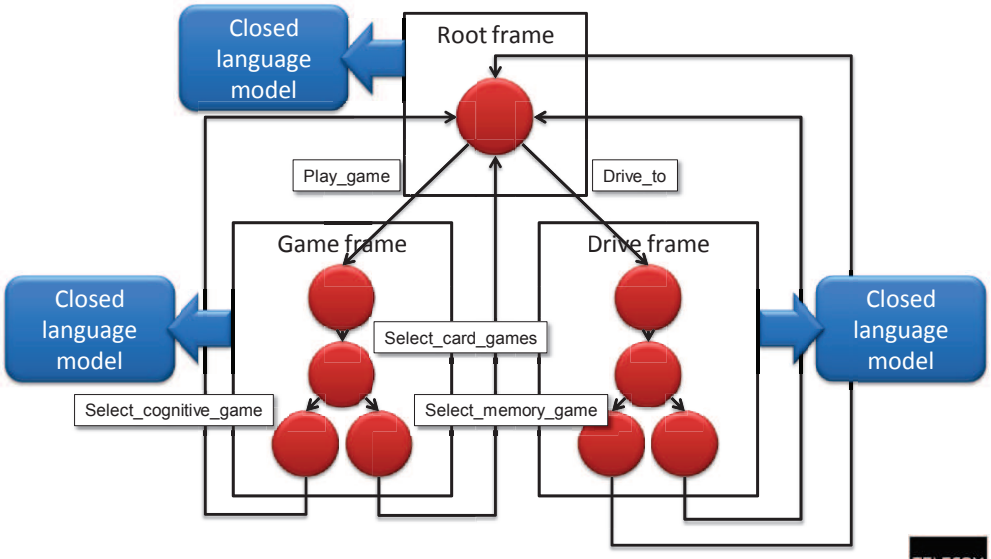
Four Steps toward the continuous listening



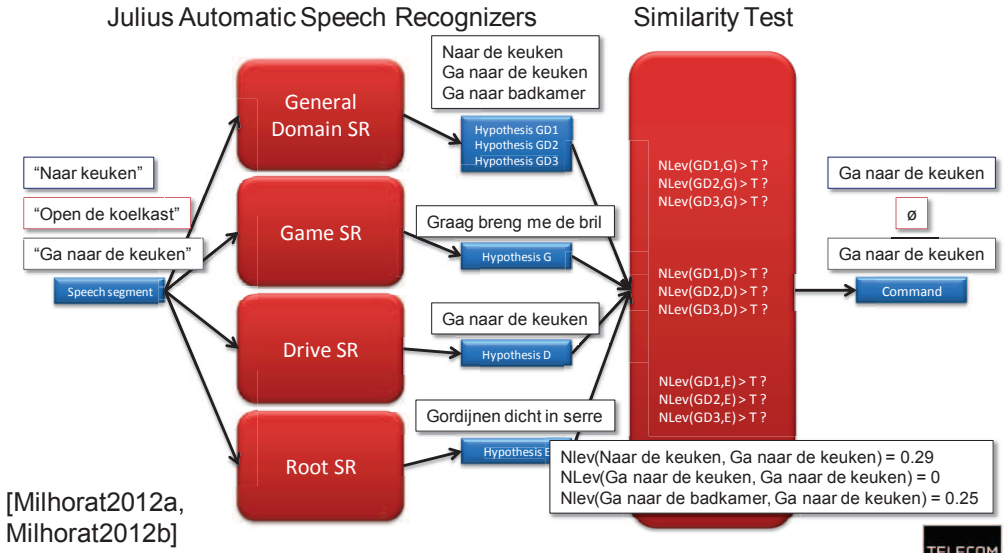
Mobile robot mounted ASR system



Mobile robot mounted ASR system



Mobile robot mounted ASR system



[Milhorat2012a, Milhorat2012b]

Normalized Levenshtein distance

- **Word-level Levenshtein distance: minimum edit distance between the Hyp word sequence and the Ref word sequence**
- **Normalized with the average count of words**

$$\mathrm{NLev}(\mathrm{Ref}, \mathrm{Hyp}) = \frac{\mathrm{Lev}(\mathrm{Ref}, \mathrm{Hyp})}{\frac{\mathrm{word\_count}(\mathrm{Ref}) + \mathrm{word\_count}(\mathrm{Hyp})}{2}}$$

- **NLev("I want to send a message", "Can you send a message") = 3/(6+5)/2 = 0.55**

Evaluation corpus

- **5 Dutch speakers**
- **50 phonetically balanced sentences for acoustic adaptation (10 each)**
- **240 in-situ test utterances:**
  - 100 in-application commands (20 each)
  - 110 out-of-application commands (22 each)
  - 30 partial commands (6 each)

Evaluation metrics

- **Sentence error rate:**

$$100 \times \frac{\text{count(badly recognized sentences)}}{\text{count(sentences)}}$$

- **Sentence false-positive rate:**

$$100 \times \frac{\text{count(wrongly validated sentences)}}{\text{count(sentences)}}$$

Evaluation

System	Sentence error rate	Sentence false-positive rate
In-applications commands		
Baseline	85% (±7%)	-
Similarity test	15% (±7%)	0%
Out-of-applications commands		
Baseline	91% (±5%)	-
Similarity test	0% (rejected)	0%
Partial commands		
Baseline	83% (±13%)	-
Similarity test	33% (±16%)	0%

A sub-system to map natural-language utterances to situated parameterized dialog acts

Task description

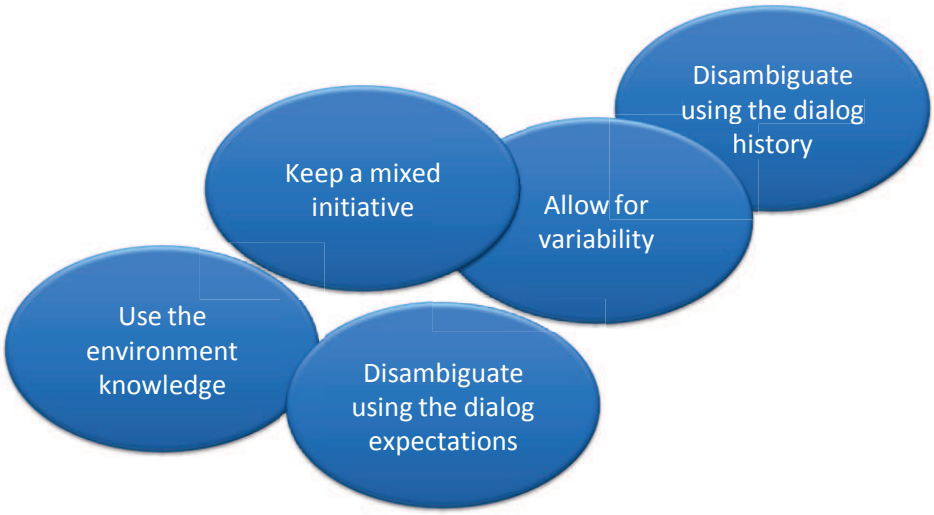
- From the wide informal natural language to the restricted world of the dialog manager

State of the art in Semantic Parsing/Natural Language Understanding

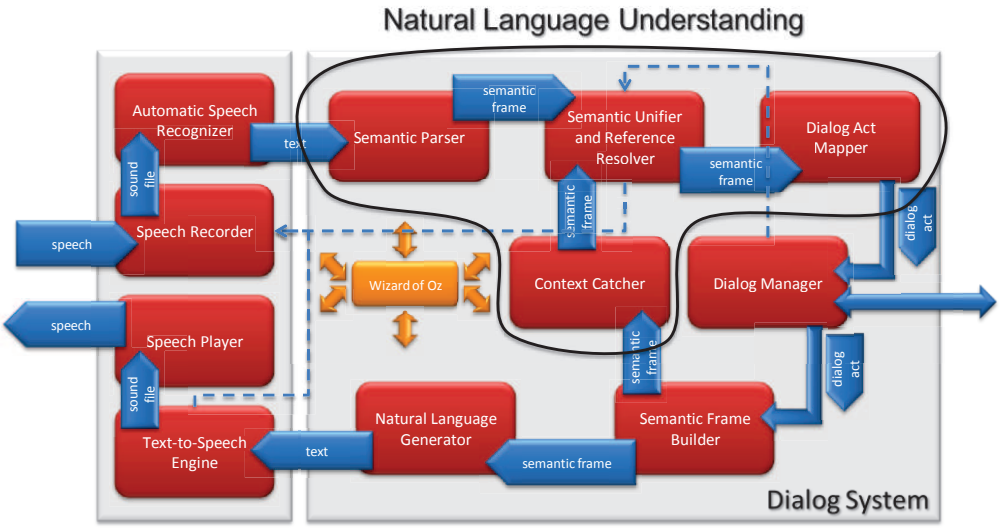
Deterministic methods	Stochastic methods
Keyword spotting [Weizenbaum1966] : looks for patterns/keywords	Hidden Markov models [Pieraccini1991] : states are semantic labels and observations are words
Context-free grammar [Ward1994] : builds parse trees from derivation rules	Hidden vector state [He2003] : parse tree as a sequence of vector states
Probabilistic context-free grammar [Gorin1997] : builds parse trees from probabilistic derivation rules	

Conclusion: investigate a multi-step NLU subsystem

Requirements

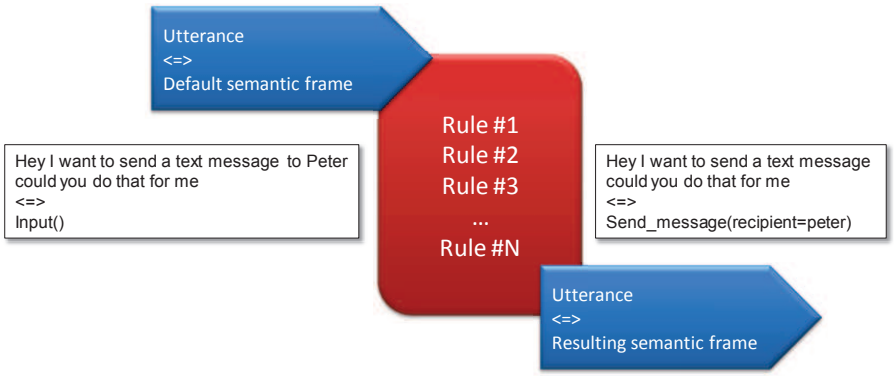


Overview

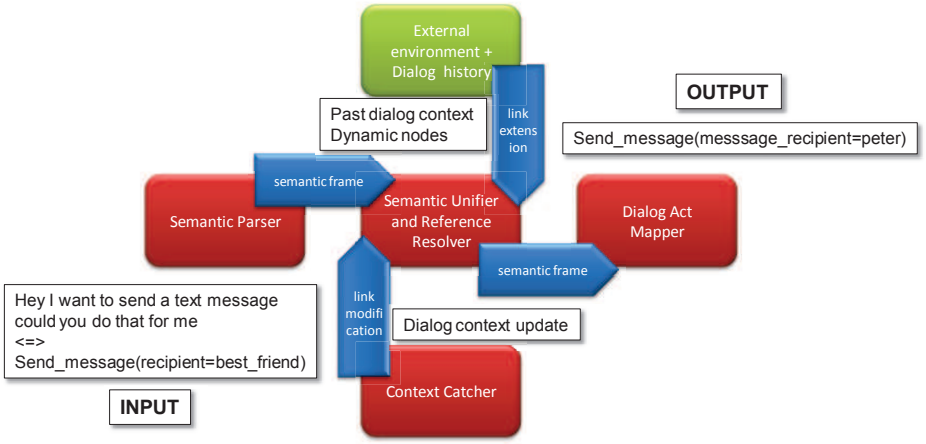


Semantic parsing [Jurcicek2009]

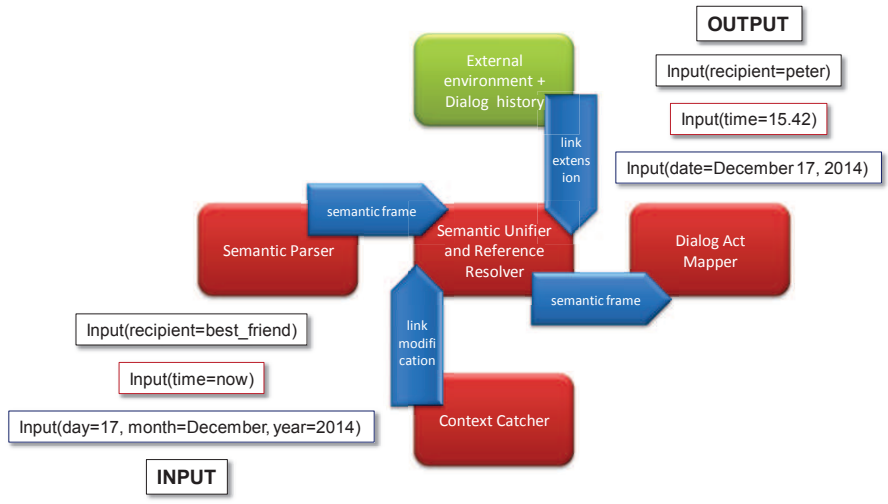
- Based on a sequence of transformations
- Use a slot database



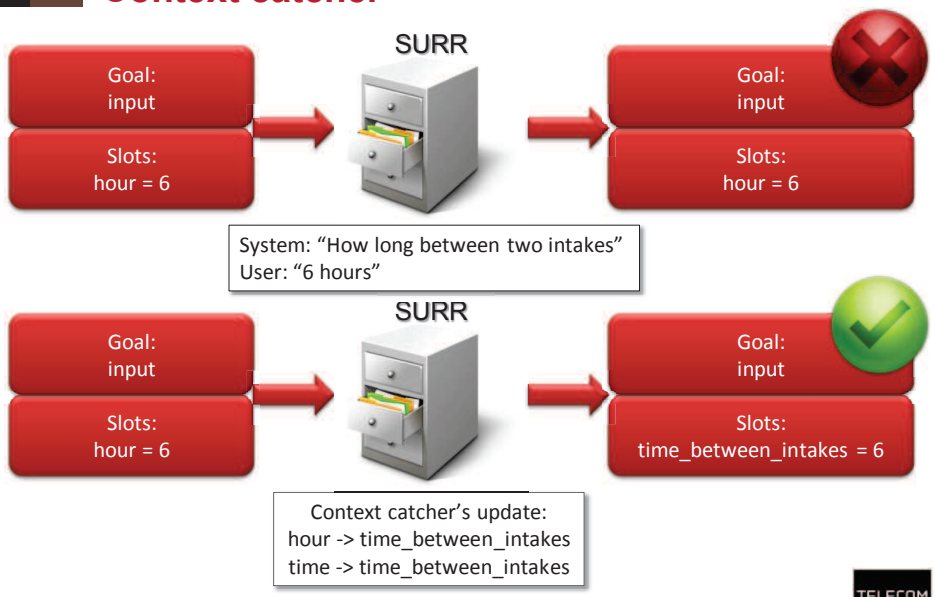
Semantic unifier and reference resolver



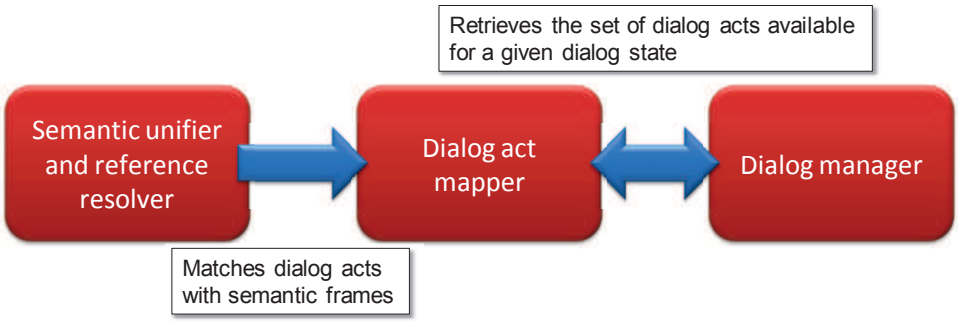
Semantic unifier and reference resolver



Context catcher



Dialog act mapper



Error recovery

- **Gradual error levels:**

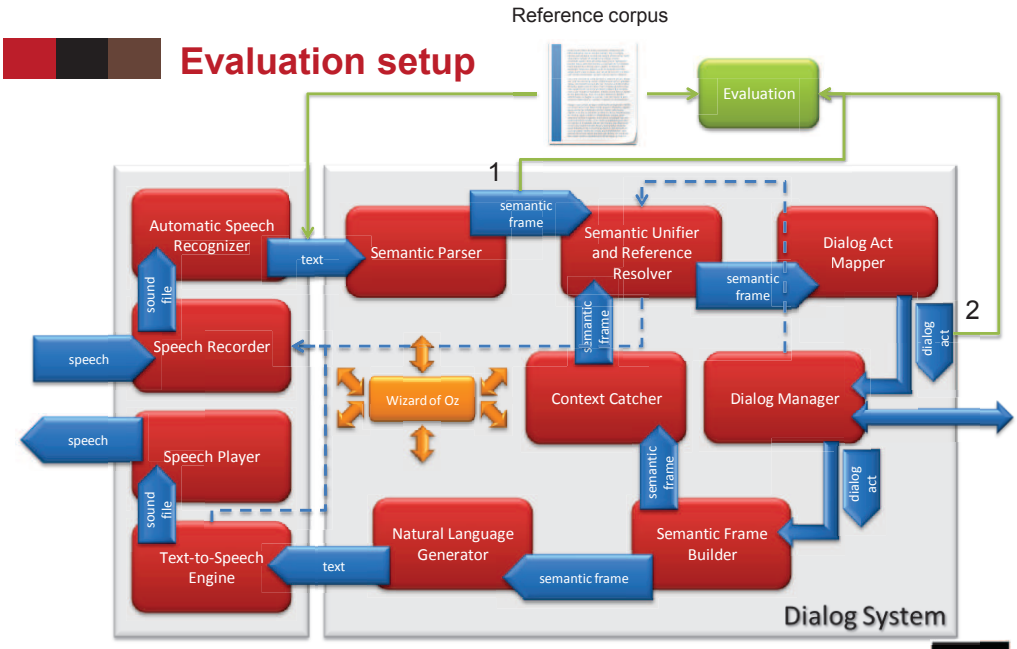
System: "What is your mood today?" User: "I want to cook a meal"		
Error level	Behavior	Examples
1	Signal that a non-understanding occurred	"I did not get that" "What did you say?"
2	Signal that a non-understanding occurred including the best ASR hypothesis	"I understood "I want to cook a meal" but I am not able to handle that request"
3	Signal that a non-understanding occurred and propose some utterances templates	"I can't handle that request. Try to say "good" or "bad", or say "start over""

Evaluation corpus

- **253 dialogs (in French and German) collected with a Wizard-of-Oz setup (WebWOZ)**
- **939 turns**

Best hypothesis	*à 9h du matin*
Semantic frame	*input:hour=9;*
Contextualized semantic frame	*input:first_intake=9h00*
Dialog act	*input:first_intake=9h00*
System's dialog act	*Ask.What:goal=frequency_per_day; slot=frequency_per_day;*
System's response	*Quelle est la fréquence des prises?*

Evaluation setup



Evaluation metrics

- **Precision:**

$$100 \times \frac{\text{count(corr ectly retrieved elements)}}{\text{count(elem ents to be retrieved)}}$$

- **Slot recall:**

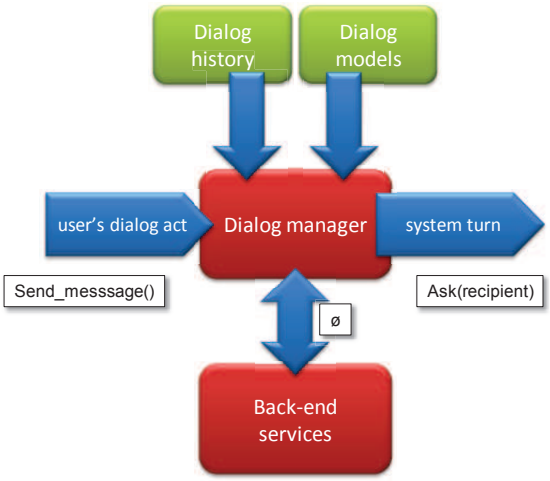
$$100 \times \frac{\text{count(corr ectly retrieved slots)}}{\text{count(slot s to be retrieved)}}$$

Overall evaluation results

Element	Precision ratio	Precision	Recall
Semantic frame (output of the semantic parser)	882 / 939	94% (±1.5%)	-
Slots (output of the semantic parser)	502 / 537	93% (±1.6%)	88% (474 / 537)
Dialog act	912 / 939	97% (±1%)	-
Dialog acts: 3% (26) incomplete, 0.1% (1) incorrect			

Linked-form filling: a new paradigm to create and update task-based dialog models

Managing dialogs

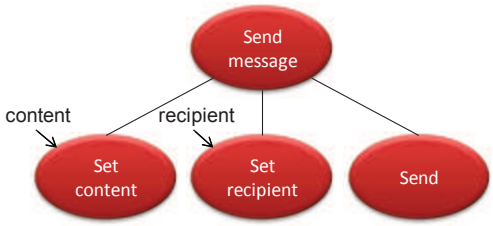


State of the art in dialog management

Deterministic methods		Stochastic methods
Adjacency pairs [Weizenbaum1966]: predefined triggers and responses	Flow graphs: finite-state machine	Probabilistic IS [Lison2012]: probabilistic update rules
		Markov decision processes [Levin1998,Levin2000]: finite-state machine with probabilistic transitions
Information state [Larsson1998]: state is the sum of the information exchanged	Example-based [Lee2006]: example database learned from corpus	Partially observable Markov decision processes [Young2006,Lemon2007]: integrates the uncertainty of the observations

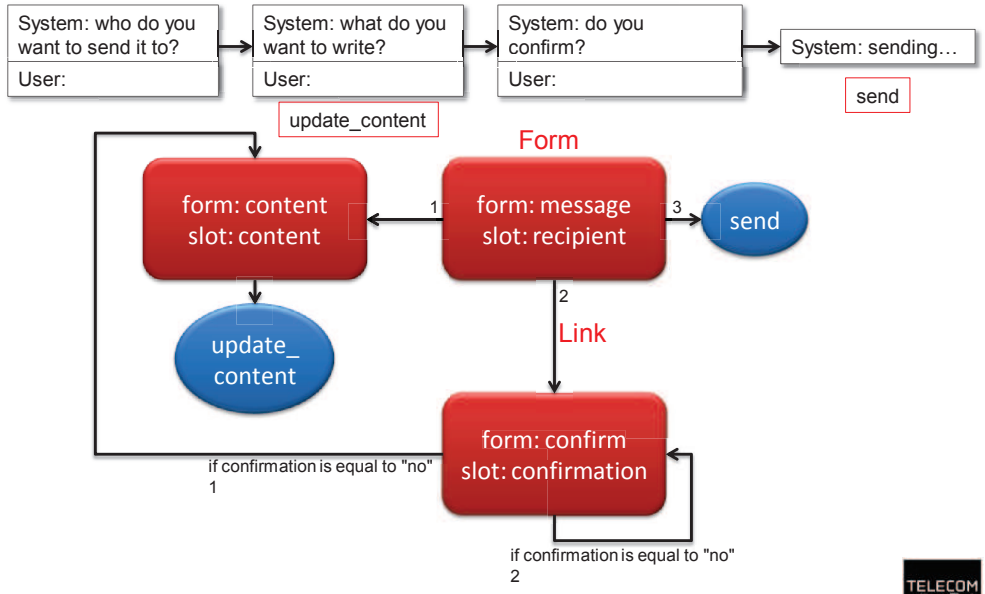
Modeling with task hierarchies

- A task is completed when all of its subtasks, if any, are completed [Bohus2003,Rich2009]
- Define and organize unit tasks

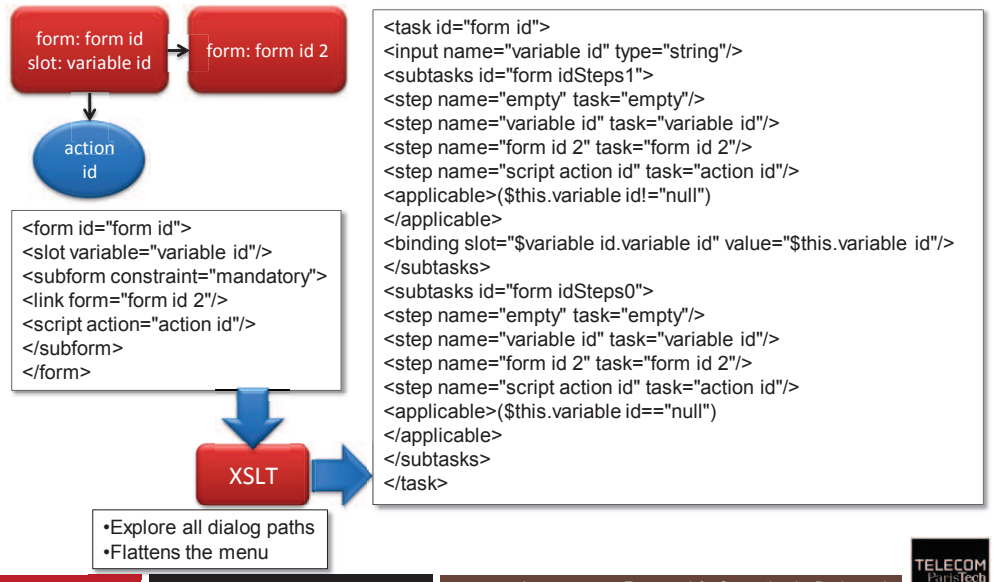


Pros	Cons
■ No data required ■ Limited time to deployment ■ Standardized modeling (ANSI/CEA-2018) ■ Available runtime tools: Disco, RavenClaw	■ Increasingly complex with the dialog variations ■ Difficult to maintain and update the dialogs (expertise, manual modeling) ■ Static structure

The linked-form filling: a language and a tool



eXtensible Stylesheet Language Transformations



Characteristics

- No data
- Facilitated design
- Quick conception
- Low expertise requirements
- Guarantee of dialog completion
- Direct domain enlargement
- Automatic dialog variability

	Task hierarchy	Linked-form filling	Reduction
Lines	139 (49-246)	30 (14-46)	76% (±5%)
XML elements	87 (24-164)	18 (7-32)	76%(±10%)

Compared to…

✓Has the characteristic
× Does not have the characteristic
- Not relevant

	No data	Facilitated design	Quick conception	Low expertise requirements	Guarantee of dialog completion	Direct domain enlargement	Automatic dialog variability
Flow charts	✓	-	✓	✓	✓	✓	×
Example-based	×	-	×	✓	×	×	✓
Information state	✓	✓	✓	×	×	×	✓
Adjacency pairs	✓	-	✓	✓	×	✓	✓
Stochastic models	×	✓	×	×	✓	×	✓
Linked form-filling	✓	✓	✓	✓	✓	✓	✓

Conclusions and future work

Conclusions

- **A modular open-source platform for spoken dialog systems**
  - Open-source platform for researchers
  - Independent components (modular, extendable, multi-modal, multi-lingual)
  - Facilitated setup (minimum data requirement)
  - Real-users deployment (server-based)

Conclusions

- **Towards continuous listening for spoken interaction**
  - Continuous method for a target speaker and environment (no press-to-speak button, attention selection, automatic segmentation)

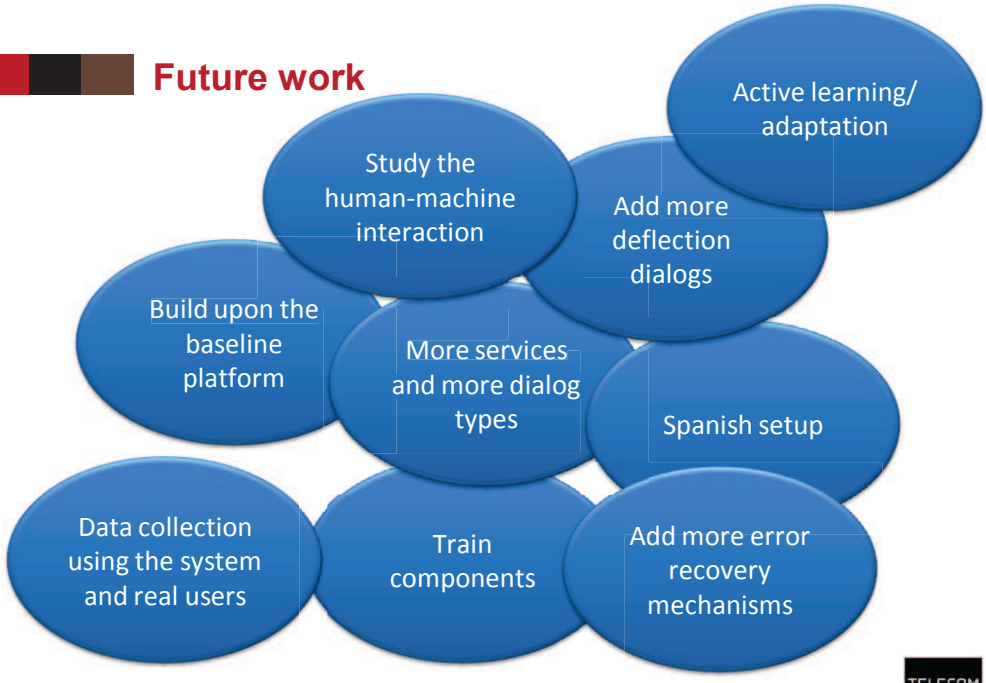
Conclusions

- **A sub-system to map natural-language utterances to situated parameterized dialog acts**
  - Handles variability
  - Allows for a mixed intiative interaction
  - Integrates the dialog context (current and past)
  - Integrates knowledge about the environment
  - Multi-lingual and portable: French, German, Italian, Spanish, etc.

Conclusions

- **Linked form-filling: a new paradigm to create and update task-based dialog models**
  - Novel paradigm
  - Quick conception
  - Low expertise requirements
  - No data required
  - Automatic transformation
  - Compliant to the ANSI/CEA-2018 standard

Future work



Publications

1. Hands-free Speech-sound Interactions At Home. In European Signal Processing Conference, 2012
2. Interactions Sonores Et Vocales Dans l'Habitat. In Interactions Langagières pour personnes Âgées Dans les habitats Intelligents, Journées d'Etudes sur la Parole, 2012
3. Un Système De Dialogue Vocal Pour Les Seniors: Etudes Et Spécifications. In Journées d'Etude sur la Télésanté, 2013
4. What If Everyone Could Do It? A Framework For Easier Spoken Dialog System Design. In Engineering Interactive Computing Systems, 2013
5. Using Wizard of Oz To Collect Interaction Data For Voice Controlled Home Care And Communication Services. In Signal Processing, Pattern Recognition and Applications, 2013
6. Designing , Building and Evaluating Voice User Interfaces For The Home. In Conference on Human Factors in Computing Systems, 2013
7. Multi-step Natural Language Understanding. In SIGdial Meeting on Discourse and Dialogue, pages 157-159, Metz, France, August 2013. Association for Computational Linguistics
8. Building the Next generation of Personal Digital Assistants. In ATSIP International Conference on Advanced Technologies for Signal & Image Processing, 2014
9. Etude des HMM paramétriques pour la reconnaissance de parole en environnement bruite. In Journées d'Etudes sur la Parole, 2014
10. Building The Personal Assistant For Dependent People - Helping Dependent People to Cope With Technology Through Speech Interaction. In HEALTHINF International Conference on Health Informatics, 2014
11. Designing Language Technology Applications: A Wizard of Oz Driven Prototyping Framework. In EACL Conference of the European Chapter of the Association for Computer Linguistics, 2014
12. Un Système de Dialogue Vocal Comme Agent d 'Aide a la Personne. In Journées d'Etude sur la TéléSanté, 2014
13. Assessing Voice User Interfaces: The vAssist System Prototype. In International Conference on Cognitive Infocommunications, 2014

References

- [Eskenazi2008] Maxine Eskenazi, Alan W. Black, Antoine Raux, and Brian Langner. Let's Go Lab : a platform for evaluation of spoken dialog systems with real world users. In Annual Conference of the International Speech Communication Association, 2008.
- [Raux20003] Antoine Raux, Brian Langner, Alan W. Black, and Maxine Eskenazi. LET'S GO: Improving Spoken Dialog Systems for the Elderly and Nonnatives. In European Conference on Speech Communication and Technology, 2003.
- [Raux2005] Antoine Raux, Brian Langner, and Dan Bohus. Let's go public! taking a spoken dialog system to the real world. In InterSpeech, 2005.

References

- [Lison2012] Pierre Lison. Declarative design of spoken dialogue systems with probabilistic rules. In Workshop on the Semantics and Pragmatics of Dialogue, pages 32-39, 2012.
- [Bohus2007] Dan Bohus, Antoine Raux, Thomas K Harris, Maxine Eskenazi, and Alexander I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, number April, pages 32-39, 2007.

References

- [Seneff1998] Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. GALAXY-II: a reference architecture for conversational system development. ICSLP, 1998.
- [McTear1998] Mickael McTear. Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. ICSLP, 1998

References

- [Weizenbaum1966] Joseph Weizenbaum. ELIZA|a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1):36{45, 1966.
- [Ward1994] Wayne Ward and Sunil Issar. Recent improvements in the CMU spoken language understanding system. In Workshop on Human Language Technology, 1994.
- [Gorin1997] Allen L. Gorin, Giuseppe Riccardi, and Jeremy H. Wright. How may I help you? Speech Communication, 1997.

References

- [Pieraccini1991] Roberto Pieraccini, Esther Levin, and Chin-hui Lee. Stochastic Representation of Conceptual Structure in the ATIS Task. In Speech and Natural Language, volume 04, pages 121-124, 1991.
- [He2003] Yulan He and Steve Young. Hidden vector state model for hierarchical semantic parsing. International Conference on Acoustics, Speech and Signal Processing, 2003.

References

- [Jurcicek2009] Filip Jurcicek, Francois Mairesse, Milica Gasic, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. Transformation-based Learning for semantic parsing. Annual Conference of the International Speech Communication Association, pages 2719-2722, 2009.
- [Larsson1998] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. Natural language engineering, September 1998.

References

- [Lee2006] Cheongjae Lee, Sangkeun Jung, Jihyun Eun, Minwoo Jeong, and Gary Geunbae Lee. A situation-based dialogue management using dialogue examples. In International Conference on Acoustics, Speech and Signal Processing, pages 69-72, 2006.
- [Levin1998] Esther Levin, Roberto Pieraccini, and Wieland Eckert. Using Markov Decision Process for Learning Dialogue Strategies. In International Conference on Acoustics, Speech and Signal Processing, pages 201{204. IEEE, 1998.

References

- [Levin2000] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. IEEE Transactions on Speech and Audio Processing, 8(1):11-23, 2000.
- [Lemon2007] Oliver Lemon and Olivier Pietquin. Machine learning for spoken dialogue systems. Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07), pages 2685-2688, 2007.

References

- [Bohus2003] Dan Bohus and Alexander I. Rudnicky. RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda. In Eurospeech, 2003.
- [Rich2009] Charles Rich. Building task-based user interfaces with ANSI/CEA-2018. Computer, 2009.
- [Young2006] Steve Young. Using POMDPs for Dialog Management. IEE/ACL Workshop on Spoken Language Technology, 2006.

An Open-source Framework for Supporting the Design and Implementation of Natural-language Spoken Dialog Systems

Pierrick MILHORAT

RESUME : L'interaction vocale avec des systèmes automatiques connaît, depuis quelques années, un accroissement dans l'intérêt que lui porte tant le grand public que la communauté de la recherche. Cette tendance s'est renforcée avec le déploiement des assistants vocaux personnels sur les terminaux portables.

Cette thèse s'inscrit dans ce cadre pour aborder le sujet depuis deux points de vue complémentaires. D'une part, celui apparent de la fiabilité, de l'efficacité et de l'utilisabilité de ces interfaces. D'autre part, les aspects de conception et d'implémentation sont étudiés pour apporter des outils de développement aux concepteurs plus ou moins initiés de tels systèmes.

A partir des outils et des évolutions dans le domaine, une plate-forme modulaire de dialogue vocal a été agrégée. Progressivement, celle-ci a été configurée pour répondre aux exigences des scénarios d'usage et de démonstration dans l'optique des collaborations encadrant ce travail. Le système s'est complexifié et est constamment en évolution suivant les approches mentionnées plus haut.

L'interaction continue, basée sur une "écoute" permanente du système pose des problèmes de segmentation, de débruitage, de capture de son, de sélection des segments adressés au système, etc... Une méthode simple, basée sur la comparaison des résultats de traitements parallèles a prouvé son efficacité, tout comme ses limites pour une interaction continue avec l'utilisateur.

Les modules de compréhension du langage forment un sous-système interconnecté au sein de la plate-forme. Ils sont les adaptations d'algorithmes de l'état de l'art comme des idées originales. Ils ont été pensé pour rendre l'interaction naturelle et fiable tout en limitant la complexité de leur configuration et en maintenant leur généricité et donc leur usage à travers plusieurs dialogues. L'utilisabilité est évaluée à partir de données collectées lors d'essais en laboratoire avec des utilisateurs réels. L'aisance dans la configuration d'un tel système et sa modularité, plus difficiles à prouver empiriquement, sont discutées.

Le choix de la gestion du dialogue basé sur des modèles de tâches hiérarchiques, comme c'est la cas pour la plate-forme, est argumenté. Ce formalisme est basé sur une construction humaine et présente, de fait, des obstacles pour concevoir, implémenter, maintenir et faire évoluer les modèles. Pour parer à ceux-ci, un nouveau formalisme est proposé qui se transforme en hiérarchie de tâches grâce aux outils associés. Ce document se veut être une référence du nouveau langage code et de sa conversion, il présente également des mesures d'évaluation de l'apport d'un tel outil.

ABSTRACT : Recently, global tech companies released so-called virtual intelligent personal assistants. Highlighting what was the emerging trend in the interaction with machines, especially on hand-held devices.

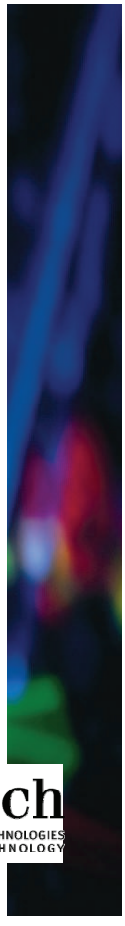
This thesis has a bi-directional approach to the domain of spoken dialog systems. On the one hand, parts of the work emphasize on increasing the reliability and the intuitiveness of such interfaces. On the other hand, it also focuses on the design and development side, providing a platform made of independent specialized modules and tools to support the implementation and the test of prototypical spoken dialog systems technologies.

The topics covered by this thesis are centered around an open-source framework for supporting the design and implementation of natural-language spoken dialog systems. The framework has been developed and set up following the use cases of the supporting projects. It is still currently evolving.

One way to characterize a spoken dialog system is by using the listening method it is applying. Continuous listening, where users are not required to signal their intent prior to speak, has been and is still an active research area. Two methods are proposed here, analyzed and compared.

According to the two directions taken in this work, the natural language understanding subsystem of the platform has been thought to be intuitive to use, allowing a natural language interaction. It is easy to set up as well since one does not need much knowledge of the technologies involved to configure the subsystem for one's application.

Finally, on the dialog management side, this thesis argue in favor of the deterministic modeling of dialogs. However, such an approach requires intense human labor, is prone to error and does not ease the maintenance, the update or the modification of the models. A new paradigm, the linked-form filling language, offers to facilitate the design and the maintenance tasks by shifting the modeling to an application specification formalism.



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY