Universidad de Granada

Programa Oficial de Posgrado en Ciencias de la Computación y Tecnología Informática (P36 56 1)

EDUARDO EISMAN

Emotions, conversational systems and heterogeneous data sources

Editor: Universidad de Granada. Tesis Doctorales
Autor: Eduardo Manuel Eisman Cabeza
ISBN: 978-84-9125-447-8
URI: http://hdl.handle.net/10481/41976

Emotions, conversational systems and heterogeneous data sources

Eduardo Manuel Eisman Cabeza

Department of Computer Science and Artificial Intelligence

University of Granada

A thesis submitted for the degree of

Doctor of Computer Science and Information Technology

Granada, October 2015

This page intentionally left blank

The memory entitled "Emotions, conversational systems and heterogeneous data sources", presented by the doctoral candidate Eduardo Manuel Eisman Cabeza to aim for the degree of Doctor of Computer Science and Information Technology, has been carried out within the "*Master in Soft Computing and Intelligent Systems*", belonging to the "*Official Doctorate Program in Computer Science and Information Technology (P36 56 1)*" of the Department of Computer Science and Artificial Intelligence (DECSAI) of the University of Granada (Spain), under the direction of professor Juan Luis Castro Peña.

The doctoral candidate and the thesis director guarantee, on signing this doctoral thesis, that the work has been made by the doctoral candidate under the direction of the thesis director and as far as our knowledge reaches, in the realization of the work, the rights of other authors to be cited have been respected, when their results or publications have been used.

The doctoral candidate has carried out a four months research stay in the Centre of Computational Intelligence of the De Montfort University in Leicester (United Kingdom) under the supervision of professor Francisco Chiclana. According to the current legislation for obtaining the International Doctorate Mention, the abstract, the introduction and the conclusions of the research are presented in two languages, Spanish and English, whereas the remaining chapters are presented just in English, the most habitual language for the scientific communication in the knowledge field to which this research work belongs.

Granada, October 2015

The doctoral candidate | The advisor

Sgd: Eduardo Manuel Eisman Cabeza | Sgd: Juan Luis Castro Peña

La memoria titulada "*Emociones, sistemas conversacionales y fuentes de datos heterogéneas*", presentada por el doctorando Eduardo Manuel Eisman Cabeza para optar al grado de Doctor en Ciencias de la Computación y Tecnología Informática, ha sido llevada a cabo dentro del "*Máster en Soft Computing y Sistemas Inteligentes*", perteneciente al "*Programa Oficial de Posgrado en Ciencias de la Computación y Tecnología Informática (P36 56 1)*" del Departamento de Ciencias de la Computación e Inteligencia Artificial (DECSAI) de la Universidad de Granada (España), bajo la dirección del catedrático Juan Luis Castro Peña.

El doctorando y el director de la tesis garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección del director de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

El doctorando ha realizado una estancia de cuatro meses en el Centre of Computational Intelligence de la De Montfort University en Leicester (Reino Unido) supervisada por el profesor Francisco Chiclana. De acuerdo a la normativa vigente para la concesión de la Mención de Doctorado Internacional, el resumen, la introducción y las conclusiones de la investigación se presentan en dos idiomas, español e inglés, mientras que el resto de capítulos se presenta únicamente en inglés, el idioma más habitual para la comunicación científica en el campo de conocimiento en el que se enmarca este trabajo de investigación.

Granada, Octubre 2015

El doctorando | El director

Fdo.: Eduardo Manuel Eisman Cabeza | Fdo.: Juan Luis Castro Peña

This page intentionally left blank

To my family.

This page intentionally left blank

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Technology under research project TIN2007-67984-C02-01, the regional Andalusian Government under research project TIC-P06-01424, a FPU research and teaching fellowship from the Spanish Ministry of Education and Science, and a Torres Quevedo research contract from the Spanish Ministry of Science and Innovation.

And last but not least, I would like to thank my thesis and stay advisors, my colleagues, my teachers, my friends and my family, for all their support along these years.

This page intentionally left blank

Abstract

The aim of this thesis is to provide solutions to three real problems demanded by information society, within the field of conversational agents and the world of natural language processing, which we have faced in different projects. First of all, the improvement of the naturalness of this kind of systems. This shortcoming reduces their credibility and it can pose a big problem in certain applications. In second place, the problem of information overload. We are in a time when more and more information is stored online. However, many times this is not completely well organized or its volume makes impossible that it can be accessed efficiently. In third place, the access to heterogeneous data sources. Nowadays, information is not usually stored in just one location, but it is scattered across various sources that need to be integrated, and each could have a different format and structure.

Along this thesis, we will study each one of these problems in depth. We will analyze the different alternatives available in literature and we will provide solutions in the form of methodologies, frameworks and concrete systems. The contributions made have been published on several journals and congresses both national and international, and the developed systems are commercially available.

Abstract (Spanish)

La finalidad de esta tesis es aportar soluciones a tres problemas reales demandados por la sociedad de la información, dentro del ámbito de los agentes conversacionales y el mundo del procesamiento de lenguaje natural, a los que nos hemos enfrentado en diferentes proyectos. En primer lugar, la mejora de la naturalidad de este tipo de sistemas. Esta limitación disminuye su credibilidad y puede suponer un gran problema en determinadas aplicaciones. En segundo lugar, el problema de la sobrecarga de información. Estamos en una época en la que cada vez se almacena más información online. Sin embargo, muchas veces esta no está del todo bien organizada o su volumen hace imposible que se pueda acceder a ella de forma eficiente. En tercer lugar, el acceso a fuentes de datos heterogéneas. Lo normal hoy en día es que la información no esté almacenada en una sola localización, sino que esté repartida por diversas fuentes que necesiten ser integradas, y cada una de las cuales podría tener un formato y una estructura diferente.

A lo largo de esta tesis, estudiaremos en profundidad cada uno de estos problemas. Analizaremos las distintas alternativas disponibles en la literatura y aportaremos soluciones en forma de metodologías, entornos de trabajo y sistemas concretos. Las contribuciones realizadas han sido publicadas en diversas revistas y congresos de carácter nacional e internacional, y los sistemas desarrollados se encuentran disponibles comercialmente.

This page intentionally left blank

Contents

List of Figures

List of Tables

This page intentionally left blank

1

Introduction

In this chapter, we will make an introduction to the different problems related to conversational systems that are tackled in this thesis: the improvement of their naturalness, the information overload, and the access to heterogeneous data sources. In addition, we will present a little state of the art of each of these three problems and we will set out a series of objectives that we tried to achieve with the realization of this thesis.

1.1 Introduction

Conversational agents are intelligent systems, usually represented by a real character or a cartoon, which are able to engage in a certain natural language conversation with a human being, interact with their environment, and behave as a real person would do. The main objective of this technology is to make human-computer interaction easier. It can be used in any situation where there is a communication among people, with the agent playing the role of any of them. Nowadays, conversational agents are being more and more used in different application domains. From agents that simulate the behavior of real people for

training on different fields, to systems that act like natural language query interfaces to large information sources. In short, an endless number of areas such as culture, entertainment, tourism, e-learning, e-commerce, or medicine, can greatly benefit from using this technology.

From the users' point of view, many studies (Chai et al., 2001b; Chai et al., 2001c; Kaufmann and Bernstein, 2010; Zhou et al., 2012) reveal that there is a clear preference for natural language interfaces to the detriment of others more traditional like keywords search of classic search engines, formal query languages, or menu driven interaction. In addition, it has been demonstrated that the interest of users decreases exponentially with the increase in the number of mouse clicks (Huberman et al., 1998), fact that is emphasized even more if we talk about mobile devices where traditional input interfaces are very limited. However, natural language systems do are able to ease and improve the user experience.

If we focus on the economical point of view, facts like the increase of e-commerce spending suppose a great opportunity for natural language systems. According to comScore [1] (one of the most important Internet marketing research companies), Q1 2014 saw desktop-based U.S. retail e-commerce spending rise 12% year-over-year to $56.1 billion. M-commerce spending on smartphones and tablets added $7.3 billion for the quarter, up 23% vs. year ago, for a digital commerce spending total of $63.4 billion in the first quarter. In addition, some events like Alibaba's Singles' Day [2] confirm that m-commerce is more and more important. During that day, the world's biggest online retail sales day [3], all predictions were exceeded. $9.3 billion in sales and 278 million orders shipped, 43% of which were placed on mobile devices. This change in consumer behavior reflects the necessity of defining more intelligent ways of interacting with websites and databases that substitute the traditional keywords search.

This thesis has been done within the official master "*Soft Computing and Intelligent Systems*", belonging to the official PhD program "*Computer Science and Information Technology*" (P36 56 1) of the Department of Computer Science

[1] https://www.comscore.com/Insights/Press-Releases/2014/5/comScore-Reports-56-1-Billion-in-Q1-2014-Desktop-Based-US-Retail-ECommerce-Spending-Up-12-Percent-vs-Year-Ago

[2] http://www.alibaba.com/

[3] http://www.bbc.com/news/business-29999289

and Artificial Intelligence (DECSAI) [1] of the University of Granada [2] (Spain). It has been done under the direction of professor Juan Luis Castro Peña.

Due to its nature, this is not an exclusively theoretical thesis, but it has a strong practical component. In fact, it provides solutions to three real problems demanded by information society, within the field of conversational agents and the world of natural language processing, which we have faced in different projects. First of all, the improvement of the naturalness of this kind of systems. This shortcoming reduces their credibility and it can pose a big problem in certain applications like the ones we will talk about later on. In second place, the problem of information overload. We are in a time when more and more information is stored online. However, many times this is not completely well organized or its volume makes impossible that it can be accessed efficiently. In third place, the access to heterogeneous data sources. Nowadays, information is not usually stored in just one location, but it is scattered across various sources that need to be integrated, and each could have a different format and structure. Next, we will detail a little bit more each of these three problems that have originated this thesis.

The problem of the improvement of naturalness is motivated by the increasing interest in the development of intelligent systems that simulate the behavior of human beings. These agents must behave with a high realism level, so they must be able to emotionally react to the events that happen in the world they live. This highlights the need for an emotional state control system that could guarantee that agents behave in a consistent way and adapt themselves to the different situations naturally.

This situation arose in the research projects "*Intelligent systems for the development of software virtual patients. TIN2007-67984-C02-01. 10/2007-09/2010*" of the Spanish Ministry of Science and Technology, and "*Virtual patient for training and e-training in medicine. P06-TIC-1424. 04/2007-03/2010*" of the Ministry of Economy, Innovation and Science of the regional Andalusian Government. Both were performed under the direction of professor Juan Luis Castro Peña and

[1] http://decsai.ugr.es/
[2] http://www.ugr.es/

with the collaboration of the *IAVANTE Foundation* [1] and the *CITIC Foundation* [2]. The objective of these projects was the development of a Virtual Simulated Patient that could be used as a tool for training and e-training the students of medicine in primary health care. It intended to solve the problem that arose from the way the training of the health personnel was done. A person was usually trained to play the role of a patient in front of students. However, this simulating technique needed human actors that played the role of patients, and it was an expensive process that had to be repeated for each new actor and health process to simulate. With the development of a Virtual Simulated Patient, we would have the same objectives and advantages as with real patients, but opening a new world of possibilities because it would not need people playing the role of patients, and it would make possible training professionals on different subjects, fitting to each one's necessities and without taking a risk for patients.

In these projects, we needed a system that was able to control the way of feeling of the patient, how she was affected by her environment, the illnesses that she suffered from, the interaction with the personnel that was being trained, and so on. In addition, this emotional state should be reflected not only on the way of feeling of the patient, but also on how she expressed and showed the symptoms she suffered from. In general, anything that could make the Virtual Simulated Patient behave more naturally, which would have a direct influence over the credibility of the patient and hence over the effectiveness of the training received by students.

The second problem that is addressed in this thesis is related to the information overload that there is nowadays on the Internet. There are certain websites which, in spite of having a good structure, organization and design, handle such amount of information that sometimes it is complicated to find a certain information in which we can be interested.

An example of this kind of websites is the one of the University of Granada. According to the Ranking Web of Universities [3], made every six months since 2004 by the Cybermetrics Lab of the Spanish National Research Council (CSIC) [4], the

[1] http://www.iavante.es/
[2] http://www.citic.es/
[3] http://www.webometrics.info/
[4] http://www.csic.es/

University of Granada is ranked, in the last report of July 2015, in places 195 worldwide and 6 nationwide, only behind universities so important such as the University of Valencia [1] (96), the Complutense University of Madrid [2] (140), the University of Barcelona [3] (145), the Autonomous University of Barcelona [4] (180) and the Polytechnic University of Catalonia BarcelonaTech [5] (189). However, and despite these good results, the existence of a large number of websites in the university about undergraduate and postgraduate studies, faculties, departments, services, vice chancellors' offices..., each with its own structure, makes really difficult accessing the information in an efficient and practical way, specially for those people who access the website for the first time (for example, the new students).

In this sense, it was evident the necessity of defining a new mechanism for organizing and accessing the information that was highly efficient and effective. In this way, users could access in real time the information that they were looking for in each moment, without having to waste their time until finding it. It is here where virtual assistants have proven to be the most effective technology in recent years. Unlike traditional search engines which do not provide immediate information and are usually limited to the use of keywords, assistants are able to interpret and answer complex natural language questions. Even, they can offer related information and serve as a guide for certain processes such as presenting procedures online.

Finally, the third problem that we have faced is the access to heterogeneous data sources. Nowadays there are more and more applications that integrate the information from different origins or services in order to provide a solution to a problem. However, many times the way of accessing that information turns out to be complicated, not very natural and inefficient for users.

This situation happens in many of the e-commerce portals that we can find nowadays, as it could be the one of a bookshop. We could identify two types of information sources in it. On the one hand, the contained in the products

[1] http://www.uv.es/
[2] http://www.ucm.es/
[3] http://www.ub.edu/
[4] http://www.uab.cat/
[5] http://www.upc.edu/

database, books in this case, with information related to titles, authors, publishers, categories, prices, publication dates, and so on. On the other hand, all the information related to the website itself about the localization of the bricks and mortar store, the opening hours, the telephone number, the payment methods, the shipping or return costs, and so on. However, these portals do not usually have any integrated system for querying the information in natural language independently of where it comes from. What is more, most of the times the access to the products catalog must be done in a completely manual way, specifying a set of filters one by one, and leaving a more personalized treat aside like the advising that we could get from the owner of the bookshop when we looked for a book with certain features in a real store.

In order to solve this problem, we needed a natural language system that could be able to integrate, transparently for the user, the knowledge existing in several independent data sources, each of which could have a different format and structure. It had to be interactive, since sometimes the requests of users turn out to be a little bit vague in the sense that they need to be completed little by little with additional information, which could be perfectly provided in a dialog with the system. That is why it was important that it had some memory and dialog management modules that could keep track of the conversation and use the context in an appropriate way. Likewise, it was essential that it could handle fuzzy concepts such as "*cheap*" or "*recent*" and temporal queries that involve relative dates, very usual in any conversation. Finally, as we have already said, the system had to be able to advise users according to their preferences.

1.2 State of the art

In this section we will make a brief review of the state of the art of each of the problems that we want to solve in this thesis. On later chapters, we will perform a more detailed analysis to show the advantages and disadvantages of the different existing alternatives related to these problems.

1.2.1 Emotions in embodied conversational agents

There are several models in literature for representing the emotional state of a conversational agent. Some (Yanaru et al., 1994) follow the idea presented by Ekman (1984) in his basic emotions theory. This asserts that there is a set of basic emotions common for every culture. Although it depends a lot on the application domain, authors usually represent emotions as vectors in an eight-dimensional space where each emotional attribute takes a real value between 0 and 1. Another alternative is the circumplex model of affect proposed by Russell and Feldman Barrett (1999). Instead of using a sequence of basic discrete and mutually excluding categories, this is based on a continuous model on the valence and arousal dimensions which eases the transition between emotions. However, it makes difficult to include new types of emotions and it does not allow the appearance of more than one basic emotion if these are not correlatives.

For controlling the emotional state of the agent there are several approaches. Many are based on the OCC theory (Ortony et al., 1988), which considers that emotions are good or bad responses of agents to events, taking their own goals and motivations into account. The difference between these approaches lies in the components that they take into account when designing the system. Bartneck (2002) simplifies the 22 emotional categories of the OCC model, too complex for the development of believable conversational agents. *FLAME* (El-Nasr et al., 2000) is a fuzzy system where emotions decay over time and an emotional filter is established. Rosis et al. (2003) use dynamic belief networks. They take account of personality, emotions overlapping, decay and filtering. *Cathexis* (Velásquez, 1997) uses a distributed architecture that also considers the temporal decay. Prendinger and Ishizuka (2001) propose an agents architecture for modeling and filtering emotions based on the social context, a similar idea to the one proposed by Elliott (1992).

Besides the structure of the emotional state control system, it is necessary an updating mechanism. Yanaru et al. (1994) make the transitions between states by means of fuzzy inference and nine rules that take account of the agent's current emotional state and the word or input term. *FLAME* (El-Nasr et al., 2000) also uses a fuzzy rules system, but it is based on the OCC model, so the rules do

not calculate the emotional state variations directly. Egges et al. (2003) use a lineal approach with two functions, one that depends on the agent's personality and the previous emotional state, and another that represents an internal change that decreases the emotional state.

Analyzing all the requirements of our problem, we identified a list of components and features to take into account in the development of a control system for the emotional state of a conversational agent: realism, gradualness, personality, emotional filtering, temporal decay, reuse, adaptability, interpretability, non-determinism, emotional stability, memory, conduct types, agents' interaction, agent's health... In this way, although there are some emotional state control systems, non-covered or improvable necessities made evident the importance of developing a new system that could solve our problem with total accuracy.

1.2.2 Closed-domain virtual assistants

As far back as the mid-nineties, several authors noticed the big growth that Internet was having and pointed out the necessity for some sort of intelligent system to help users who surfed looking for a certain information. First systems could be considered more as tools, since they did not allow asking natural language questions. For example, Letizia (Lieberman, 1995) was an agent for web browser which analyzed the user's behavior and suggested pages of interest from a set of heuristics. WebWatcher (Armstrong et al., 1995) used a vector of word occurrences to represent the pages, the links and the goals of the user. When she did a search within a web page, the system highlighted some links on it. In addition, it recommended links as the user entered new pages. Yan et al. (1996) proposed a system which analyzed the pages access logs to make groups of similar users that could be used to improve the organization of websites or suggest links dynamically. WebView (Cockburn et al., 1999) was a plug-in for the Netscape navigator which included features such as miniatures of visited pages, links to navigate directly to subordinate pages, organization of pages by temporal properties or structural relationships... Little by little these systems were gaining popularity, like Microsoft's agents (Ball et al., 1997), interactive cartoons that made it easy the use of applications and web pages, although sometimes they were too invasive

and they did not show a very complex behavior.

Moreover, systems that used a little bit more advanced techniques and algorithms such as case-based reasoning or neural networks were appearing. In Hypercase (Micarelli and Sciarrone, 1996) a set of experts on the domain predefined all possible thematic paths through the web, which were grouped by goals, so that it could be found the closest to the user's behavior. Broadway (Jaczynski and Trousse, 1998) also used case-based reasoning. Depending on the visited pages, it was able to infer the objectives of users and propose potentially relevant recommendations. Other approaches were based on the use of graphs. Footprints (Wexelblat and Maes, 1999) were a set of tools that took advantage of the work made by previous users. In this case, the pages were the nodes of the graph and the links were the transitions. The pages were annotated with the percentage of users who had followed each link, and the transitions created by users revealed paths of how the pages should be connected.

This type of systems kept evolving and including features similar to the ones that we can find in conversational agents or virtual assistants nowadays. REA (Cassell et al., 2000) was a real estate salesperson who interacted with users to determine their needs. Her body was projected on a screen on top of which some cameras acting as sensors were installed, and she included automatic speech recognition and text-to-speech. SAMIR (Abbattista et al., 2004) was an agent for an online bookstore that was based on a custom version of ALICE [1] which included a 3D face and a classifier to keep the conversation and the facial expressions coherent with each other. This kind of agents based on AIML (Wallace, 2003) use pairs of patterns and templates specified by hand, so that the agent answers the template associated to the pattern that best matches the question. The problem is that every time a web page is updated, it is necessary to modify the related rules. In order to avoid this, some authors like Kimura and Kitamura (2006) represented the dynamic content of pages by means of RDF and used SPARQL queries for the agents to speak based on those contents.

In turn, they started to work more and more at the integration with external systems. CHATTIE (Kim et al., 2005) was an assistant oriented to e-commerce which was easy to integrate with information retrieval and database management

[1] http://www.alicebot.org/aiml/aaa/

systems. It used several agents: one for simple dialog queries, another for frequently asked context-independent questions, and another for context-dependent dialogs. NaVir (Khentout et al., 2007) made the learning process easy in a platform for distance education. Pilato et al. (2008) proposed an intelligent tutoring system to learn Java which used a hierarchical ontology to store the concepts and their relations. Each document had a frequent words list associated to it which could be used to make groups and evaluate the knowledge of students. In that way, when a student asked a question, the system looked for the concept that best matched it and created a backward path to the student's current knowledge state. In addition, we could find some commercial assistants on the Web such as the ones of Artificial Solutions [1].

In short, many of these systems, specially those from the first years, are pure tools that do not allow the use of natural language. They usually lack a virtual character able to show emotions that make the interaction process friendlier. In addition sometimes the information provided by the domain, the context, and the memory to answer precisely is wasted. Other times, it is not allowed to answer general domain questions, in several languages, or generate dynamic answers whose content depends on certain restrictions. Moreover, the behavior of these systems is usually set beforehand and it is difficult to adapt. The information or the recommendations provided are sometimes limited by the content stored in the database itself, and some systems do not allow guiding the user through a certain process either. For all these reasons, we proposed a methodology and a framework for the design of closed-domain virtual assistants which could be integrated into every existing website and would cover these and other mentioned features.

1.2.3 Accessing heterogeneous data sources

Depending on how the knowledge is structured, natural language interfaces (NLIs) are mainly divided in three categories. NLIs to structured data, which are usually closed-domain and can work either on relational databases, translating the query to SQL (the so called Natural Language Interfaces to Databases), or on ontologies, translating the query to SPARQL. NLIs to non-structured or semi-structured

[1] http://www.artificial-solutions.com/

data, which are usually open-domain and process huge amounts of documents to find the answer to a question. Finally, interactive NLIs, which are used in dialog systems and have memory to remember previous questions and restrictions. We will focus on NLIs to structured data and interactive NLIs.

Although there exist other ways of classification, natural language interfaces to databases are usually classified according to their architecture (Androutsopoulos et al., 1995; Pazos R. et al., 2013). In this way, we can find four types of systems: pattern matching, syntax-based, semantic grammars, and intermediate representation languages.

Pattern matching was the typical architecture of first systems, although it has also been used later in systems like gNarLI (Shankar and Yung, 2000). They basically use patterns (simple templates which are searched in the user's input to identify the type of question) and expressions (queries, usually SQL, associated to the kind of question and used to get the desired information from the database). Their main advantage is their simplicity. They are really easy to implement, and they do not need any syntactic analysis. In addition, they sometimes provide reasonable answers to questions that are out of the range of sentences which the patterns were designed for. However, they present a huge disadvantage, their superficiality. They make many mistakes and the coverage is limited by the number of patterns.

Syntax based systems like LUNAR (Woods et al., 1972) (a NLIDB about chemical analyses of Apollo-11 moon rocks) analyze the sentence to make a syntactic tree which is translated into predicate logic to get the database query. They use lexicons (catalogs of words of the language) and grammar rules (combinations of words to form meaningful sentences in that language). Their main advantage is that they provide a detailed structure of the sentence that can be easily mapped to the database query. Nevertheless, they present some drawbacks such as semantic ambiguity, that there may be several interpretations for a unique query, bad portability, and that it is not always clear when a node should add semantic information.

Semantic grammar systems, such as LADDER (Hendrix et al., 1978), PLANES (Waltz, 1978), or PRECISE (Popescu et al., 2003), are very similar to syntax based systems. They simplify the parse tree to the minimum, removing

or combining the nodes that are not needed for making a semantic interpretation of the query. Their main advantage is their great performance. They do not need complex syntactic trees. In addition, the semantic information is assigned to tree nodes, so they reduce the problems when users omit some words. However, these systems are difficult to port. New semantic grammars are needed for new domains, although some systems like WASP (Wong and Mooney, 2006) try to make these rules automatically from a corpus or interacting with the user. Nevertheless, (semi) automatic approaches also require a great amount of work to annotate the corpus before it can be used.

Intermediate representation languages, instead of directly translating the question into SQL using a syntax-based approach, they use an intermediate language of logic queries (predicate logic), which is then translated into a specific database query language. For example, CHAT-80 (Warren and Pereira, 1982) transformed questions about world geography into Prolog expressions. Their main advantages are the independence of the database management system and the possibility of including reasoning modules between the semantic analyzer and the database query generator. Nevertheless, these systems have a limited application scope, since most of them use deductive databases, much less used than the relational ones.

Regarding interactive natural language interfaces, we can find several systems that perform a dialog and memory management. For example, JUPITER (Zue et al., 2000) was an over-the-telephone conversational interface that allowed obtaining weather forecast information for over 500 cities worldwide. It included a table with geographical information organized hierarchically that allowed summarizing too lengthy results. MERCURY (Seneff and Polifroni, 2000) provided information over the telephone about itineraries and prices of flights involving the 150 busiest airports worldwide. It entailed additional challenges such as time zone differences or geographical proximity of airports. The system included a fuzzy matching heuristic, and users were permitted to update constraints at any point in time. Natural Language Assistant (NLA) (Chai et al., 2002) helped to find information about products and services in an e-commerce site. It included a statistical parsing that allowed scaling to multiple languages easily, and a rule-based system with weights to implement business rules. In addition, it initiated dialogs

to ask for additional information, and generated explanations for recommendations. FREyA (Damljanovic et al., 2012) was oriented to querying ontologies and the Linked Open Data [1] and it also used clarification dialogs to train the system and improve its performance in time.

With regard to purely commercial systems, one of the most popular is Anna [2], the IKEA's automatic assistant. She is available in 16 languages, also includes memory and answers a great amount of general domain questions (personal information, hobbies...) and specific knowledge about IKEA (products, opening hours, payment methods, shipping costs...).

In spite of the initial popularity of first systems, in general these were gradually disappearing due to several problems: bad performance, significant effort to develop specialized systems for individual databases that could not be easily adapted to other domains, and so on. After some years in the oblivion, this type of systems started to live a second youth. However, if we focus on the broader concept of natural language interface that can be used by any kind of public (not only expert users or database administrators) and is able to integrate different information sources simultaneously and transparently for users, there are still many problems to solve and many aspects to improve.

The majority of systems are focused on a very specific part of the problem. Some like NaLIR (Li and Jagadish, 2014b) have interesting features as NLIs. They support complex SQL queries, but they use just one knowledge source and lack conversational capabilities. Others, like ORAKEL (Cimiano et al., 2008), focus on minimizing the effort of adapting the system to a given domain. On the other hand, systems like Natural Language Assistant (Chai et al., 2002) do include a dialog manager, but they neither support general domain queries nor work with different types of data sources at the same time. In the case of a commercial system like Anna, she is much more complete in that conversational side, but not so advanced in the interaction with the database (she does not handle fuzzy or temporal queries), and she does not seem to include a full emotional model. These and other problems such as the cleansing and filtering of information from the database, the use of taxonomies to enrich that information, or the inclusion

[1] http://linkeddata.org/
[2] http://www.ikea.com/us/en/

of an advising module to advise users in a personalized way, are the ones that we try to solve with the design of a new system for accessing heterogeneous data sources.

1.3 Objectives

The main objective of this thesis is to provide solutions to a series of interesting open problems related to the domain of conversational systems that we consider fundamental: the improvement of their naturalness, the information overload, and the access to heterogeneous data sources. Based on that, we can specify a set of general objectives:

- To make an emotion modeling system that is easy to interpret and modify, and improves the naturalness of conversational agents. Our initial hypothesis is that the application of fuzzy rule-based systems would provide a good solution to this problem.

- To make easy the access, in an immediate and precise way, to great amounts of dynamic information related to a certain domain. We believe that the use of semantic structures such as ontologies and the application of rules and restrictions during the decision making process would allow organizing and taking advantage of all the existing knowledge in the domain.

- To combine different heterogeneous data sources to allow users to access information in an integrated, transparent, effective, efficient, and pleasant way. We consider that the use of multi-agent systems able to combine expert agents and decision agents would allow solving this problem in a modular and scalable way.

Moreover, in order to achieve these objectives not only theoretically, we have considered to develop systems of commercial interest which put the acquired knowledge and the designed models into practice, and can be used to solve real problems:

1. To develop an emotional state control system for conversational agents which determines their way of acting, behaving as a real person would do, and includes the following features:

   (a) Ability to reflect a great amount of emotions: joy, disdain, anger, fear, worry, surprise, sadness, and embarrassment.

   (b) Capability to include different types of personality.

   (c) Different intensity levels for each emotion.

   (d) Configurable, interpretable, and justifiable behavior from the human being's point of view, without black box models that neither it is very well known how they work, nor why they behave in a certain way.

   (e) Environment and self-agent perception influenced by the emotional state of the same.

   (f) Emotional answers determined by the current state of the agent.

   (g) Certain emotional stability which prevents emotions from appearing or disappearing too quickly.

   (h) Representation by means of a 2D or 3D model which presents a high realism level.

   (i) Non-deterministic behavior.

2. To develop a conversational system which allows accessing, in an immediate and precise way, a great amount of dynamic information related to a certain organization, including the following requirements:

   (a) Effective, taking advantage of the information provided by the domain to answer the questions precisely.

   (b) Definition of information units based on a class hierarchy with different properties.

   (c) Dynamic answers whose content may depend on some patterns, temporal variables, or any other type of restriction.

(d) Complementary to the web, providing information that there is no reason to be contained in the same, but it is only used as support when informing.

(e) Possibility of answering general domain questions.

(f) Behavior configurable by means of rules.

(g) Use of context and memory in conversation.

(h) Recommendations based on the connections between the different information units.

(i) Efficient, capable of answering in real time.

(j) Multilingual, understanding and answering questions in different languages.

(k) Possibility of including automatic speech recognition and text-to-speech modules.

(l) Ability to guide the user throughout a certain process defined in stages.

(m) Definition of different levels of restricted access to information.

(n) Capability to offer all the information related to an object in a summarized way.

(o) Compatible with the emotional state control system.

3. To develop a conversational system which allows accessing heterogeneous data sources simultaneously and includes the features that are detailed next:

   (a) Information retrieval and integration from different data sources simultaneously.

   (b) Semantic recognition of questions and treatment of different linguistic phenomena.

   (c) Efficient and effective, capable of answering questions precisely and in real time.

   (d) Preprocessing of the information stored in the database, including data cleansing and filtering systems.

(e) Interactive system with memory and dialog management modules which allow users to complete their requests little by little with additional information.

(f) Advising module to advise users in a personalized way.

(g) Inclusion of category hierarchies (taxonomies) to enrich the information from the database.

(h) Recognition of fuzzy concepts such as "*cheap*" or "*recent*".

(i) Recognition of temporal queries which involve relative dates.

(j) Ease of adaptability to different domains.

(k) Multilingual, understanding and answering questions in different languages.

(l) Possibility of including automatic speech recognition and text-to-speech modules.

(m) Recommendation of related information.

(n) Compatible with the emotional state control system.

1.4 Methodology

In order to achieve the proposed objectives, a common methodology has been followed for the three problems to solve:

- To make an introduction explaining the motivation of each of the problems.
- To analyze the different proposals existing in literature, identifying their advantages and disadvantages.
- To set a list of functional and quality requirements regarding such aspects of conversational systems.
- To determine if analyzed proposals fulfill all specified requirements and they are enough to solve the posed problems.
- To propose alternatives which add value in those cases where it is necessary.

- To design generic models which provide solutions to each problem.
- To develop commercial systems that apply those generic models to the resolution of specific problems.
- To evaluate the performance and reliability of the implemented systems.
- To provide a series of conclusions and highlight some possible directions for future work.

This methodology is reflected along the next chapters.

1.5 Thesis structure

This memory is organized in three big chapters which are focused on each of the main contributions of this thesis. Each chapter has been written in a self-contained way, that is, it has been included the enough information with the necessary references so that it may be read more or less independently. Nevertheless, it is recommended to follow the order specified in this memory since it reflects how all work has been carried out.

In this way, after the introduction made in Chapters 1 and 2 (English and Spanish, respectively), Chapter 3 presents a system for controlling the emotional state of a conversational agent, which is easily adaptable to different applications domains, highly interpretable, and able to provide a certain emotional stability. Next, Chapter 4 presents a framework for designing closed-domain virtual assistants which allows solving efficiently the problems of information overload, disorganization, and lack of accessibility in websites. Afterwards, Chapter 5 shows a generic multi-agent architecture for conversational systems that allows accessing heterogeneous data sources simultaneously, which are the kind of problems that are usually found nowadays in which the information is scattered across different origins, each with a particular structure and format. Later, Chapters 6 and 7 (English and Spanish) include a series of conclusions and show some lines for future work. Finally, Chapter 8 show a list of publications which reflects the work done in this thesis.

1.6 Research stays

Some of the work included in this memory has been carried out during a four months stay (June 1, 2009 – September 30, 2009) at the Centre of Computational Intelligence of the De Montfort University in Leicester (United Kingdom), financed by a FPU research and teaching fellowship from the Spanish Ministry of Education and Science, and under the supervision of professor Francisco Chiclana.

This page intentionally left blank

2

Introduction (Spanish)

En este capítulo, realizaremos una introducción a los diferentes problemas relacionados con los sistemas conversacionales que se abordan en esta tesis: la mejora de su naturalidad, la sobrecarga de información y el acceso a fuentes de datos heterogéneas. Asimismo, presentaremos un pequeño estado del arte de cada uno de estos tres problemas y plantearemos una serie de objetivos que pretendíamos alcanzar con la realización de esta tesis.

2.1 Introducción

Los agentes conversacionales son sistemas inteligentes, normalmente representados por un personaje real o animado, capaces de mantener una cierta conversación en lenguaje natural con un ser humano, interactuar con su entorno y comportarse como lo haría una persona real. El principal objetivo de esta tecnología es facilitar la interacción hombre-máquina. Puede utilizarse en cualquier situación en la que exista una comunicación entre personas, con el agente desempeñando el papel de cualquiera de ellas. En la actualidad, los agentes conversacionales están siendo cada vez más utilizados en diferentes dominios de aplicación. Desde agentes que

simulan el comportamiento de personas reales para la formación en diferentes ámbitos, hasta sistemas que actúan como interfaces de consulta en lenguaje natural a grandes fuentes de información. En definitiva, un sinfín de áreas como la cultura, el entretenimiento, el turismo, el aprendizaje, el comercio electrónico o la medicina pueden beneficiarse enormemente del uso de esta tecnología.

Desde el punto de vista de los usuarios, muchos estudios (Chai y col., 2001b; Chai y col., 2001c; Kaufmann y Bernstein, 2010; Zhou y col., 2012) revelan que realmente existe una clara preferencia por las interfaces en lenguaje natural en detrimento de otras más tradicionales como la búsqueda por palabras clave de los buscadores clásicos, los lenguajes de consulta formales, o la interacción mediante menús. Además, se ha demostrado que el interés de los usuarios disminuye exponencialmente a medida que aumenta el número de clics de ratón (Huberman y col., 1998), hecho que se acentúa aún más si hablamos de dispositivos móviles en los que las interfaces de entrada tradicionales están muy limitadas. Sin embargo, los sistemas basados en lenguaje natural sí que son capaces de facilitar y mejorar la experiencia de usuario.

Si nos centramos en el punto de vista económico, hechos como el incremento del gasto en comercio electrónico suponen una gran oportunidad para los sistemas basados en lenguaje natural. Según comScore [1] (una de las compañías más importantes en investigación de mercado en Internet), en el primer trimestre del año 2014, en Estados Unidos se produjo un incremento interanual del 12 % en las ventas al por menor de comercio electrónico, alcanzando los 56.100 millones de dólares. Las ventas del comercio móvil desde teléfonos inteligentes y tabletas añadieron 7.300 millones de dólares al trimestre, un 23 % más que el año anterior, haciendo un total de 63.400 millones de dólares en ese primer trimestre. Además, algunos eventos como el Día del Soltero de Alibaba [2] confirman que el comercio móvil es cada vez más importante. Durante ese día, el de mayores ventas a través de Internet a nivel mundial [3], se superaron todos los pronósticos. 9.300 millones de dólares en ventas y 278 millones de pedidos enviados, el 43 % de los cuales se

[1] https://www.comscore.com/Insights/Press-Releases/2014/5/comScore-Reports-56-1-Billion-in-Q1-2014-Desktop-Based-US-Retail-ECommerce-Spending-Up-12-Percent-vs-Year-Ago

[2] http://www.alibaba.com/

[3] http://www.bbc.com/news/business-29999289

realizaron desde dispositivos móviles. Este cambio en el comportamiento de los consumidores refleja la necesidad de definir formas más inteligentes de interactuar con sitios web y bases de datos que sustituyan a la clásica búsqueda por palabras clave.

Esta tesis ha sido realizada dentro del máster oficial "*Soft Computing y Sistemas Inteligentes*", perteneciente al programa oficial de posgrado en "*Ciencias de la Computación y Tecnología Informática*" (P36 56 1) del Departamento de Ciencias de la Computación e Inteligencia Artificial (DECSAI) [1] de la Universidad de Granada [2]. La dirección ha sido llevada a cabo por el catedrático Juan Luis Castro Peña.

Debido a su naturaleza, esta no es una tesis exclusivamente teórica, sino que tiene una fuerte componente práctica. De hecho, aporta soluciones a tres problemas reales demandados por la sociedad de la información, dentro del ámbito de los agentes conversacionales y el mundo del procesamiento de lenguaje natural, a los que nos hemos enfrentado en diferentes proyectos. En primer lugar, la mejora de la naturalidad de este tipo de sistemas. Esta limitación disminuye su credibilidad y puede suponer un gran problema en determinadas aplicaciones como las que comentaremos más adelante. En segundo lugar, el problema de la sobrecarga de información. Estamos en una época en la que cada vez se almacena más información online. Sin embargo, muchas veces esta no está del todo bien organizada o su volumen hace imposible que se pueda acceder a ella de forma eficiente. En tercer lugar, el acceso a fuentes de datos heterogéneas. Lo normal hoy en día es que la información no esté almacenada en una sola localización, sino que esté repartida por diversas fuentes que necesiten ser integradas, y cada una de las cuales podría tener un formato y una estructura diferente. A continuación, detallaremos un poco más cada uno de estos tres problemas que han dado origen a esta tesis.

El problema de la mejora de la naturalidad viene motivado por el creciente interés en el desarrollo de sistemas inteligentes que simulen el comportamiento de los seres humanos. Estos agentes deben comportarse con un alto nivel de realismo, por lo que tienen que ser capaces de reaccionar emocionalmente a los

[1] http://decsai.ugr.es/
[2] http://www.ugr.es/

eventos que ocurren en el mundo en el que viven. Esto pone de manifiesto la necesidad de un sistema de control del estado emocional que garantice que los agentes se comporten de forma consistente y se adapten con naturalidad a las diferentes situaciones.

Este escenario es el que se presentaba en los proyectos de investigación "*Sistemas inteligentes para el desarrollo de pacientes virtuales software. TIN2007-67984-C02-01. 10/2007-09/2010*" del Ministerio de Ciencia y Tecnología, y "*Paciente virtual para formación y teleformación en medicina. P06-TIC-1424. -04/2007-03/2010*" de la Consejería de Economía, Innovación y Ciencia de la Junta de Andalucía. Ambos fueron llevados a cabo bajo la dirección de D. Juan Luis Castro Peña y con la colaboración de la *Fundación IAVANTE* [1] y la *Fundación CITIC* [2]. Estos proyectos tenían como objetivo el desarrollo de un Paciente Simulado Virtual que sirviera como herramienta de formación y teleformación en atención primaria para los estudiantes de medicina. Pretendía resolver el problema que surgía de cómo se realizaba el entrenamiento del personal sanitario. Normalmente se entrenaba a una persona para que simulase el papel de un paciente frente a los estudiantes. Sin embargo, esta técnica de simulación necesitaba actores humanos que tomasen el papel de pacientes, y era un proceso caro que debía repetirse para cada nuevo actor y proceso sanitario a simular. Con el desarrollo de un Paciente Simulado Virtual, se obtendrían los mismos objetivos y ventajas que con los pacientes reales, pero se abriría un nuevo mundo de posibilidades ya que no sería necesario que personas desempeñasen el papel de pacientes, y sería posible entrenar a los profesionales en diferentes materias, ajustándose a las necesidades de cada uno y sin suponer un riesgo para los pacientes.

Dentro de estos proyectos, necesitábamos un sistema que fuese capaz de controlar la forma de sentir del paciente, cómo le afectaba su entorno, las enfermedades que padecía, la interacción con el personal que estaba siendo entrenado, etcétera. Además, este estado emocional no sólo debía reflejarse en la manera de sentir del paciente, sino también en su forma de expresarse y mostrar los síntomas que padecía. En general, todo aquello que pudiese hacer que el Paciente Simulado Virtual se comportase con una mayor naturalidad, lo cual influiría directamente

[1] http://www.iavante.es/
[2] http://www.citic.es/

en la credibilidad del paciente y por tanto en la eficacia de la formación recibida por los estudiantes.

El segundo problema que se aborda en esta tesis está relacionado con la sobrecarga de información que existe actualmente en Internet. Existen determinados sitios web en los que, a pesar de tener una buena estructura, organización y diseño, es tal la cantidad de información que manejan que en ocasiones resulta complicado encontrar determinada información en la que podamos estar interesados.

Un ejemplo de este tipo de sitios es el de la Universidad de Granada. Según el Ranking Web de Universidades [1], elaborado cada seis meses desde 2004 por el Laboratorio de Cibermetría del Consejo Superior de Investigaciones Científicas (CSIC) [2] de España, la Universidad de Granada se sitúa, en el último informe de julio de 2015, en los puestos 195 a nivel mundial y 6 a nivel nacional, sólo por detrás de universidades tan importantes como la Universitat de València [3] (96), la Universidad Complutense de Madrid [4] (140), la Universitat de Barcelona [5] (145), la Universitat Autònoma de Barcelona [6] (180) y la Universitat Politècnica de Catalunya BarcelonaTech [7] (189). Sin embargo, y a pesar de estos buenos resultados, la existencia de gran cantidad de webs dentro de la universidad sobre estudios de grado y posgrado, facultades, departamentos, servicios, vicerrectorados..., cada una con su propia estructura, dificulta enormemente el acceso a la información de forma eficiente y práctica, especialmente para aquellas personas que acceden a la web por primera vez (por ejemplo, los alumnos de nuevo ingreso).

En este sentido, resultaba evidente la necesidad de definir un nuevo mecanismo de organización y acceso a la información que fuese altamente eficiente y efectivo. De esta forma, los usuarios podrían acceder en tiempo real a la información que buscasen en cada momento, sin tener que perder tiempo hasta encontrarla. Es aquí donde los asistente virtuales han demostrado ser la tecnología más eficaz en estos últimos años. A diferencia de los buscadores tradicionales que no proporcionan

[1] http://www.webometrics.info/
[2] http://www.csic.es/
[3] http://www.uv.es/
[4] http://www.ucm.es/
[5] http://www.ub.edu/
[6] http://www.uab.cat/
[7] http://www.upc.edu/

una información inmediata y suelen estar limitados al uso de palabras clave, los asistentes son capaces de interpretar y responder a preguntas complejas en lenguaje natural. Incluso, pueden ofrecer información relacionada y servir de guías en determinados procesos como la presentación de trámites a través de Internet.

Finalmente, el tercer problema al que nos hemos enfrentado es el acceso a fuentes de datos heterogéneas. En la actualidad cada vez existen más aplicaciones que integran la información procedente de diferentes orígenes o servicios para dar solución a un problema. Sin embargo, la forma de acceder a dicha información resulta muchas veces enrevesada, poco natural e ineficiente para los usuarios.

Esta situación se da en muchos de los portales de comercio electrónico existentes hoy en día, como podría ser el de una librería. En ella podríamos identificar dos tipos de fuentes de información. Por un lado, la contenida en la base de datos de productos, en este caso los libros, con información relativa a títulos, autores, editoriales, categorías, precios, fechas de publicación, etcétera. Por otra parte, toda la información almacenada en la propia página web acerca de la localización de la tienda física, el horario, el teléfono, los métodos de pago, los gastos de envío, devolución, etcétera. Sin embargo, estos portales no suelen contar con ningún sistema integrado de consulta de la información en lenguaje natural que sea independiente del origen de la misma. Es más, en la mayoría de las ocasiones el acceso al catálogo de productos debe realizarse de forma completamente manual, especificando uno a uno una serie de filtros, y dejando de lado un trato más personalizado como el asesoramiento que podríamos recibir del dueño de la librería en la búsqueda de un libro con unas determinadas características en una tienda real.

Para resolver este problema, necesitábamos un sistema basado en lenguaje natural que fuese capaz de integrar, de forma transparente para el usuario, el conocimiento existente en varias fuentes de datos independientes, cada una de las cuales podía tener un formato y una estructura diferente. Debía ser interactivo, ya que las peticiones de los usuarios resultan en ocasiones un poco vagas en el sentido de que necesitan completarse poco a poco con información adicional, la cual podría proporcionarse perfectamente mediante un diálogo con el sistema. Por ello era importante que este contase con unos módulos de memoria y gestión de diálogo que pudieran seguir la conversación y usar el contexto de una forma

adecuada. Asimismo, resultaba esencial que pudiese manejar conceptos difusos como "*barato*" o "*reciente*" y consultas temporales que implicasen fechas relativas, muy habituales en cualquier conversación. Finalmente, como hemos comentado, el sistema debía ser capaz de asesorar a los usuarios en base a sus preferencias.

2.2 Estado del arte

En esta sección realizaremos una breve revisión del estado del arte de cada uno de los problemas que queremos resolver en esta tesis. En capítulos posteriores, llevaremos a cabo un análisis más detallado para mostrar las ventajas e inconvenientes de las diferentes alternativas existentes relacionadas con estos problemas.

2.2.1 Emociones en agentes conversacionales

Existen varios modelos en la literatura para representar el estado emocional de un agente conversacional. Algunos (Yanaru y col., 1994) siguen la idea presentada por Ekman (1984) en su teoría de las emociones básicas. Esta sostiene que existe un conjunto de emociones básicas que son comunes para todas las culturas. Aunque depende mucho del dominio de aplicación, los autores suelen representar las emociones como vectores en un espacio de dimensión ocho donde cada atributo emocional toma un valor real entre 0 y 1. Otra alternativa es el modelo de afecto circumplejo propuesto por Russell y Feldman Barrett (1999). En lugar de utilizar una secuencia de categorías básicas discretas y mutuamente excluyentes, este se basa en un modelo continuo sobre las dimensiones de valencia y excitación que facilita la transición entre emociones. Sin embargo, dificulta la inclusión de nuevos tipos de emociones y no permite la aparición de más de una emoción básica si estas no son correlativas.

Para el control del estado emocional del agente existen varias aproximaciones. Muchas se basan en la teoría OCC (Ortony y col., 1988), la cual considera que las emociones son respuestas buenas o malas de los agentes a los eventos, teniendo en cuenta sus metas y motivaciones. La diferencia entre dichas aproximaciones radica en los componentes que tienen en cuenta a la hora de diseñar el sistema. Bartneck (2002) simplifica las 22 categorías emocionales del modelo OCC, dema-

siado complejo para el desarrollo de agentes conversacionales creíbles. *FLAME* (El-Nasr y col., 2000) es un sistema difuso donde las emociones decaen con el paso del tiempo y se establece un filtrado emocional. Rosis y col. (2003) utilizan redes de creencia dinámicas. Tienen en cuenta la personalidad, el solapamiento de emociones, su decaimiento y filtrado. *Cathexis* (Velásquez, 1997) utiliza una arquitectura distribuida que también considera el decaimiento temporal. Prendinger e Ishizuka (2001) proponen una arquitectura de agentes para modelar y filtrar emociones en base al contexto social, idea similar a la propuesta por Elliott (1992).

Aparte de la estructura del sistema de control del estado emocional, es necesario un mecanismo de actualización. Yanaru y col. (1994) realizan las transiciones entre estados mediante inferencia difusa y nueve reglas que tienen en cuenta el estado emocional actual del agente y la palabra o término de entrada. *FLAME* (El-Nasr y col., 2000) también utiliza un sistema de reglas difusas, pero está basado en el modelo OCC, por lo que las reglas no calculan directamente las variaciones en el estado emocional. Egges y col. (2003) utilizan una aproximación lineal con dos funciones, una que depende de la personalidad del agente y el estado emocional previo, y otra que representa un cambio interno que disminuye el estado emocional.

Analizando todos los requisitos de nuestro problema, identificamos una lista de componentes y características a tener en cuenta en el desarrollo de un sistema de control del estado emocional de un agente conversacional: realismo, gradualidad, personalidad, filtrado emocional, decaimiento temporal, reutilización, adaptabilidad, interpretabilidad, no determinismo, estabilidad emocional, memoria, tipos de conducta, interacción entre agentes, estado de salud del agente... Así, aunque existen muchos sistemas de control del estado emocional, las necesidades no cubiertas o mejorables pusieron de manifiesto la importancia de desarrollar un nuevo sistema que resolviese nuestro problema con total precisión.

2.2.2 Asistentes virtuales de dominio cerrado

Ya desde mediados de los 90, diversos autores se percataron del gran crecimiento que estaba teniendo Internet e identificaron la necesidad de algún tipo de sistema

inteligente para ayudar a los usuarios que navegaban buscando cierta información. Los primeros sistemas podían considerarse más como herramientas, ya que no permitían realizar preguntas en lenguaje natural. Por ejemplo, Letizia (Lieberman, 1995) era un agente para navegador web que analizaba el comportamiento del usuario y sugería páginas de interés a partir de un conjunto de heurísticas. WebWatcher (Armstrong y col., 1995) utilizaba un vector de ocurrencias de palabras para representar las páginas, los enlaces y los objetivos del usuario. Cuando este realizaba una búsqueda en una página, el sistema resaltaba ciertos enlaces en ella. Además, recomendaba enlaces a medida que el usuario accedía a nuevas páginas. Yan y col. (1996) propusieron un sistema que analizaba los registros de acceso a las páginas para crear grupos de usuarios similares que podían utilizarse para mejorar la organización de las webs o sugerir enlaces dinámicamente. WebView (Cockburn y col., 1999) era un plugin para el navegador Netscape que incluía características como miniaturas de páginas visitadas, enlaces para navegar directamente a páginas subordinadas, organización de las páginas por propiedades temporales o relaciones estructurales... Poco a poco estos sistemas fueron ganando popularidad, como los agentes de Microsoft (Ball y col., 1997), personajes animados interactivos que facilitaban el uso de aplicaciones y páginas webs, aunque en ocasiones resultaban demasiado invasivos y no mostraban un comportamiento muy complejo.

Asimismo, fueron apareciendo sistemas que utilizaban técnicas y algoritmos un poco más avanzados como el razonamiento basado en casos o las redes neuronales. En Hypercase (Micarelli y Sciarrone, 1996) una serie de expertos en el dominio predefinían todos los posibles caminos temáticos a través de la web, los cuales eran agrupados por objetivos, de forma que se podía encontrar el más cercano al comportamiento del usuario. Broadway (Jaczynski y Trousse, 1998) también utilizaba razonamiento basado en casos. En función de las páginas visitadas, era capaz de inferir los objetivos de los usuarios y proponer recomendaciones potencialmente relevantes. Otras aproximaciones se basaban en el uso de grafos. Footprints (Wexelblat y Maes, 1999) eran un conjunto de herramientas que aprovechaban el trabajo realizado por usuarios anteriores. En este caso, las páginas eran los nodos del grafo y los enlaces eran las transiciones. Las páginas eran anotadas con el porcentaje de usuarios que habían seguido cada enlace, y las transiciones creadas

por los usuarios revelaban caminos de cómo debían conectarse las páginas.

Este tipo de sistemas siguieron evolucionando e incluyendo características similares a las que podemos encontrar en los agentes conversacionales o asistentes virtuales hoy en día. REA (Cassell y col., 2000) era un agente inmobiliario que interactuaba con los usuarios para determinar sus necesidades. Su cuerpo era proyectado en una pantalla encima de la cual se instalaban unas cámaras que actuaban como sensores, e incluía reconocimiento automático del habla y generación de voz. SAMIR (Abbattista y col., 2004) era un agente para una librería online basado en una versión personalizada de ALICE [1] que incluía una cara 3D y un clasificador para mantener coherentes la conversación y las expresiones faciales. Este tipo de agentes basados en AIML (Wallace, 2003) utilizan parejas de patrones y plantillas especificadas a mano, de manera que el agente responde con la plantilla asociada al patrón que mejor casa con la pregunta. El problema es que cada vez que se actualiza una página web, es necesario modificar las reglas relacionadas. Para evitar esto, algunos autores como Kimura y Kitamura (2006) representaban el contenido dinámico de las páginas mediante RDF y utilizaban consultas en SPARQL para que los agentes hablasen en base a esos contenidos.

A su vez, se empezó a trabajar cada vez más en la integración con sistemas externos. CHATTIE (Kim y col., 2005) era un asistente orientado a comercio electrónico fácil de integrar con sistemas de recuperación de información y gestión de bases de datos. Usaba varios agentes: uno para consultas de diálogo simples, otro para preguntas frecuentes independientes del contexto, y otro para los diálogos dependientes del contexto. NaVir (Khentout y col., 2007) facilitaba el proceso de aprendizaje dentro de una plataforma de educación a distancia. Pilato y col. (2008) propusieron un sistema de tutoría inteligente para aprender Java que utilizaba una ontología jerárquica para almacenar los conceptos y sus relaciones. Cada documento tenía asociada una lista de palabras frecuentes que podía utilizarse para formar grupos y evaluar los conocimientos de los estudiantes. Así, cuando un estudiante realizaba una pregunta, el sistema buscaba el concepto que mejor casaba con ella y creaba un camino hacia atrás hasta el estado de conocimiento actual del estudiante. Además, en la Web podríamos encontrar algunos asistentes

[1] http://www.alicebot.org/aiml/aaa/

comerciales como los de Artificial Solutions [1].

En definitiva, muchos de estos sistemas, especialmente los de los primeros años, son puras herramientas que no permiten el uso de lenguaje natural. Suelen carecer de un personaje virtual capaz de mostrar emociones que haga el proceso de interacción más ameno. También en ocasiones se desaprovecha la información que aporta el dominio, el contexto y la memoria para responder de forma precisa. En otras, no se permite contestar preguntas de ámbito general, en varios idiomas, o generar respuestas dinámicas cuyo contenido dependa de ciertas restricciones. Además, el comportamiento de estos sistemas suele estar fijado de antemano y es difícil de adaptar. La información o las recomendaciones proporcionadas a veces están limitadas por el contenido almacenado en la propia web, y algunos sistemas tampoco permiten guiar al usuario a lo largo de un determinado proceso. Por todo ello, propusimos una metodología y un entorno de trabajo para el diseño de asistentes virtuales de dominio cerrado que podrían integrarse en cualquier sitio web existente y cubrirían estas y otras características mencionadas.

2.2.3 Acceso a fuentes de datos heterogéneas

Dependiendo de cómo se estructura el conocimiento, las interfaces de lenguaje natural (ILNs) se dividen principalmente en tres categorías. Las ILNs a datos estructurados, que suelen ser de dominio cerrado y pueden trabajar sobre bases de datos relacionales, traduciendo la consulta a SQL (las llamadas interfaces de lenguaje natural a bases de datos), o sobre ontologías, traduciendo la consulta a SPARQL. Las ILNs a datos no estructurados o semi-estructurados, que normalmente son de dominio abierto y procesan una gran cantidad de documentos para encontrar la respuesta a una pregunta. Por último, las ILNs interactivas, que se utilizan en sistemas de diálogo y tienen memoria para recordar preguntas y restricciones previas. Nosotros nos centraremos en las ILNs a datos estructurados e interactivas.

Aunque existen otras formas de clasificación, normalmente las interfaces de lenguaje natural a bases de datos se clasifican según su arquitectura (Androutsopoulos y col., 1995; Pazos R. y col., 2013). De esta forma, podemos encontrar

[1] http://www.artificial-solutions.com/

cuatro tipos de sistemas: búsqueda de patrones, basados en sintaxis, gramáticas semánticas, y lenguajes de representación intermedia.

La búsqueda de patrones era la típica arquitectura de los primeros sistemas, aunque también se ha usado posteriormente en sistemas como gNarLI (Shankar y Yung, 2000). Básicamente utilizan patrones (plantillas simples que se buscan en la entrada del usuario para identificar el tipo de pregunta) y expresiones (consultas, normalmente SQL, asociadas al tipo de pregunta y utilizadas para obtener la información deseada de la base de datos). Su principal ventaja es su simplicidad. Son fáciles de implementar y no necesitan ningún análisis sintáctico. Además, a veces proporcionan respuestas razonables ante preguntas que están fuera del rango de frases para los que fueron diseñados los patrones. Sin embargo, presentan una enorme desventaja, su superficialidad. Cometen muchos errores y la cobertura está limitada por el número de patrones.

Los sistemas basados en sintaxis como LUNAR (Woods y col., 1972) (una ILNBD sobre análisis químicos de rocas lunares del Apollo-11) analizan la frase para crear un árbol sintáctico que se traduce a lógica de predicados para obtener la consulta a la base de datos. Utilizan lexicones (catálogos de palabras del lenguaje) y reglas gramaticales (combinaciones de palabras para formar frases con significado en dicho lenguaje). Su principal ventaja es que proporcionan una estructura detallada de la frase que se puede traducir fácilmente a la consulta a la base de datos. No obstante, presentan algunos inconvenientes como la ambigüedad semántica, que pueda haber varias interpretaciones para una única consulta, la mala adaptabilidad, y que no siempre está claro cuándo un nodo debería añadir información semántica.

Los sistemas de gramática semántica, como LADDER (Hendrix y col., 1978), PLANES (Waltz, 1978) o PRECISE (Popescu y col., 2003), son muy similares a los basados en sintaxis. Simplifican al mínimo el árbol del análisis, eliminando o combinando los nodos que no son necesarios para realizar una interpretación semántica de la consulta. Su principal ventaja es su gran rendimiento. No necesitan complejos árboles sintácticos. Además, la información semántica está asignada a los nodos del árbol, por lo que reduce los problemas cuando los usuarios omiten ciertas palabras. Sin embargo, estos sistemas son difíciles de adaptar. Para nuevos dominios se necesitan nuevas gramáticas semánticas, aunque algunos sistemas

como WASP (Wong y Mooney, 2006) intentan construir estas reglas automáticamente a partir de un corpus o interactuando con el usuario. No obstante, las aproximaciones (semi) automáticas también requieren una gran cantidad de trabajo para anotar el corpus antes de que pueda ser usado.

Los lenguajes de representación intermedia, en lugar de traducir directamente la pregunta a SQL usando una aproximación basada en sintaxis, utilizan un lenguaje intermedio de consultas lógicas (lógica de predicados), el cual se traduce posteriormente a un lenguaje de consulta a base de datos específico. Por ejemplo, CHAT-80 (Warren y Pereira, 1982) transformaba preguntas sobre geografía mundial en expresiones Prolog. Sus principales ventajas son la independencia del sistema de gestión de bases de datos y la posibilidad de incluir módulos de razonamiento entre el analizador semántico y el generador de consultas a la base de datos. No obstante, estos sistemas tienen un ámbito de aplicación limitado, puesto que la mayoría utilizan bases de datos deductivas, mucho menos usadas que las relacionales.

En relación a las interfaces de lenguaje natural interactivas, podemos encontrar diversos sistemas que llevan a cabo una gestión de diálogo y de memoria. Por ejemplo, JUPITER (Zue y col., 2000) era una interfaz conversacional telefónica que permitía obtener información meteorológica de más de 500 ciudades del mundo. Incluía una tabla con información geográfica organizada jerárquicamente que permitía resumir los resultados que eran demasiado largos. MERCURY (Seneff y Polifroni, 2000) proporcionaba información telefónica sobre rutas y precios de vuelos de los 150 aeropuertos del mundo con mayor número de pasajeros. Implicaba retos adicionales como la diferencia de zonas horarias o la proximidad geográfica de los aeropuertos. El sistema incluía una heurística de búsqueda difusa, y los usuarios podían actualizar las restricciones en cualquier momento. Natural Language Assistant (NLA) (Chai y col., 2002) ayudaba a encontrar información sobre productos y servicios en un portal de comercio electrónico. Incluía un analizador estadístico que permitía escalar a varios idiomas fácilmente, y un sistema basado en reglas con pesos para implementar reglas de negocio. Además, iniciaba diálogos para solicitar información adicional, y generaba explicaciones para las recomendaciones. FREyA (Damljanovic y col., 2012) estaba orientado a

la consulta de ontologías y el Linked Open Data [1] y también utilizaba diálogos de clarificación para entrenar al sistema y mejorar su rendimiento con el tiempo.

Con respecto a los sistemas puramente comerciales, uno de los más conocidos es Anna [2], la asistente automática de IKEA. Está disponible en 16 idiomas, también incluye memoria y responde una gran cantidad de preguntas de propósito general (información personal, aficiones...) y de conocimiento específico de IKEA (productos, horarios de apertura, métodos de pago, gastos de envío...).

A pesar de la popularidad inicial de los primeros sistemas, en general estos fueron desapareciendo gradualmente debido a varios problemas: mal rendimiento, esfuerzo significativo para desarrollar sistemas especializados para bases de datos individuales que no se podían adaptar fácilmente a otros dominios, etcétera. Después de unos años en el olvido, este tipo de sistemas empezaron a vivir una segunda juventud. Sin embargo, si nos centramos en el concepto más amplio de interfaz de lenguaje natural que pueda ser usada por cualquier tipo de público (no sólo por usuarios expertos o administradores de bases de datos) y que sea capaz de integrar diferentes fuentes de información de manera simultánea y transparente para el usuario, todavía siguen siendo muchos los problemas a resolver y los aspectos a mejorar.

La mayoría de los sistemas se centran en una parte muy concreta del problema. Algunos como NaLIR (Li y Jagadish, 2014b) tienen características interesantes como ILNs. Permiten realizar consultas SQL complejas, pero utilizan una sola fuente de conocimiento y carecen de capacidad conversacional. Otros, como ORAKEL (Cimiano y col., 2008), tratan de minimizar el esfuerzo de adaptar el sistema a un dominio dado. Por otra parte, sistemas como Natural Language Assistant (Chai y col., 2002) sí incluyen un gestor de diálogo, pero ni permiten consultas de dominio general, ni trabajan con diferentes tipos de fuentes de datos a la vez. En el caso de un sistema comercial como Anna, es mucho más completo en esa parte conversacional, pero no tan avanzado en la interacción con la base de datos (no reconoce consultas difusas ni temporales), ni parece incorporar un completo modelo de emociones. Estos y otros problemas como los de la limpieza y el filtrado de la información procedente de la base de datos, el uso de taxonomías

[1] http://linkeddata.org/
[2] http://www.ikea.com/es/es/

que enriquezcan dicha información, o la inclusión de un módulo de asesoramiento que permita aconsejar a los usuarios de forma personalizada, son los que tratamos de resolver con el diseño de un nuevo sistema de acceso a fuentes de datos heterogéneas.

2.3 Objetivos

El principal objetivo de esta tesis es aportar soluciones a una serie de problemas abiertos y de interés relacionados con el campo de los sistemas conversacionales que consideramos fundamentales: la mejora de su naturalidad, la sobrecarga de información y el acceso a fuentes de datos heterogéneas. En base a ello, podemos especificar una serie de objetivos generales:

- Crear un sistema de modelado de emociones que sea fácil de interpretar y modificar, y mejore la naturalidad de los agentes conversacionales. Nuestra hipótesis de partida es que la aplicación de sistemas basados en reglas difusas proporcionaría una buena solución a este problema.

- Facilitar el acceso, de forma inmediata y precisa, a grandes cantidades de información dinámica relacionada con un determinado dominio. Creemos que la utilización de estructuras semánticas como las ontologías y la aplicación de reglas y restricciones en el proceso de toma de decisiones permitirían organizar y aprovechar todo el conocimiento existente en el dominio.

- Combinar diferentes fuentes de datos heterogéneas para permitir a los usuarios acceder a la información de una manera integrada, transparente, efectiva, eficiente, y atractiva. Consideramos que el uso de sistemas multiagente capaces de combinar agentes expertos y agentes de decisión permitiría resolver este problema de una forma modular y escalable.

Asimismo, para alcanzar estos objetivos no sólo de forma teórica, nos hemos planteado desarrollar sistemas de interés comercial que pongan en práctica el conocimiento adquirido y los modelos diseñados, y se puedan utilizar en la resolución de problemas reales:

1. Desarrollar un sistema de control del estado emocional para agentes conversacionales que determine su forma de actuar, comportándose como lo haría una persona real, y que incluya las siguientes características:

   (a) Capacidad de reflejar una gran cantidad de emociones: alegría, desdén, ira, miedo, preocupación, sorpresa, tristeza y vergüenza.

   (b) Capacidad de incluir diferentes tipos de personalidad.

   (c) Diferentes niveles de intensidad para cada emoción.

   (d) Comportamiento configurable, interpretable y justificable desde el punto de vista del ser humano, sin modelos de caja negra que no se sabe muy bien cómo funcionan ni por qué se comportan de una determinada manera.

   (e) Percepción del entorno y del propio agente influida por el estado emocional del mismo.

   (f) Respuestas emocionales determinadas por el estado actual del agente.

   (g) Cierta estabilidad emocional que impida que las emociones aparezcan o desaparezcan demasiado rápido.

   (h) Representación mediante un modelo 2D o 3D que presente un alto nivel de realismo.

   (i) Comportamiento no determinista.

2. Desarrollar un sistema conversacional que permita acceder, de forma inmediata y precisa, a gran cantidad de información dinámica relacionada con una determinada organización, incluyendo los siguientes requisitos:

   (a) Efectivo, aprovechando la información que aporta el dominio para contestar las preguntas de forma precisa.

   (b) Definición de unidades de información en base a una jerarquía de clases con diferentes propiedades.

   (c) Respuestas dinámicas cuyo contenido pueda depender de ciertos patrones, variables temporales, o cualquier otro tipo de restricción.

(d) Complementario a la web, proporcionando información que no tenga por qué estar contenida en la misma, sino que esta se utilice solamente como un apoyo a la hora de informar.

(e) Posibilidad de responder a preguntas de ámbito general.

(f) Comportamiento configurable mediante reglas.

(g) Utilización del contexto y la memoria en la conversación.

(h) Recomendaciones en base a las interconexiones entre las diferentes unidades de información.

(i) Eficiente, capaz de responder en tiempo real.

(j) Multilingüe, entendiendo y respondiendo a las preguntas en diferentes idiomas.

(k) Posibilidad de incorporación de módulos de reconocimiento automático del habla y síntesis de voz.

(l) Capacidad de guiar al usuario a lo largo de un determinado proceso definido por etapas.

(m) Definición de diferentes niveles de acceso restringido a la información.

(n) Capacidad de ofrecer toda la información relativa a un objeto de forma esquematizada.

(o) Compatible con el sistema de control del estado emocional.

3. Desarrollar un sistema conversacional que permita el acceso simultáneo a fuentes de datos heterogéneas e incorpore las características que se detallan a continuación:

(a) Recuperación e integración de información de diferentes fuente de datos de forma simultánea.

(b) Reconocimiento semántico de las preguntas y tratamiento de diferentes fenómenos linguísticos.

(c) Eficiente y efectivo, capaz de responder a las preguntas de forma precisa y en tiempo real.

(d) Preprocesamiento de la información contenida en la base de datos, incluyendo sistemas de limpieza y filtrado de datos.

(e) Sistema interactivo con módulos de memoria y gestión de diálogo que permitan a los usuarios ir completando sus peticiones poco a poco con información adicional.

(f) Módulo de asesoramiento que permita aconsejar a los usuarios de forma personalizada.

(g) Inclusión de jerarquías de categorías (taxonomías) para enriquecer la información de la base de datos.

(h) Reconocimiento de conceptos difusos como "*barato*" o "*reciente*".

(i) Reconocimiento de consultas temporales que incorporen fechas relativas.

(j) Facilidad de adaptación a diferentes dominios.

(k) Multilingüe, entendiendo y respondiendo a las preguntas en diferentes idiomas.

(l) Posibilidad de incorporación de módulos de reconocimiento automático del habla y síntesis de voz.

(m) Recomendación de información relacionada.

(n) Compatible con el sistema de control del estado emocional.

2.4 Metodología

Para alcanzar los objetivos propuestos, se ha utilizado una metodología común para los tres problemas a resolver:

- Realizar una introducción explicando la motivación de cada uno de los problemas.

- Analizar las diferentes propuestas existentes en la literatura, identificando sus ventajas e inconvenientes.

- Establecer una lista de requisitos funcionales y de calidad en relación a dichos aspectos de los sistemas conversacionales.
- Determinar si las propuestas analizadas cumplen los requisitos especificados y son suficientes para resolver los problemas planteados.
- Proponer alternativas que aporten valor en aquellos casos que sea necesario.
- Diseñar modelos genéricos que aporten soluciones a cada problema.
- Desarrollar sistemas comerciales que apliquen esos modelos genéricos a la resolución de problemas específicos.
- Evaluar el rendimiento y la fiabilidad de los sistemas implementados.
- Proporcionar una serie de conclusiones y destacar algunas direcciones posibles para el trabajo futuro.

Esta metodología se refleja a lo largo de los próximos capítulos.

2.5 Estructura de la tesis

Esta memoria está organizada en tres grandes capítulos que se centran en cada una de las principales contribuciones de esta tesis. Cada capítulo ha sido redactado de manera autocontenida, es decir, se ha incluido la información suficiente con las referencias necesarias para que se pueda leer de forma más o menos independiente. No obstante, se recomienda seguir el orden especificado en esta memoria ya que refleja cómo se ha ido desarrollando todo el trabajo.

Así, tras la introducción realizada en los Capítulos 1 y 2 (inglés y español, respectivamente), el Capítulo 3 presenta un sistema para el control del estado emocional de un agente conversacional, el cual es fácilmente adaptable a diferentes dominios de aplicación, altamente interpretable y capaz de proporcionar una cierta estabilidad emocional. A continuación, el Capítulo 4 presenta un entorno de trabajo para el diseño de asistentes virtuales de dominio cerrado que permite resolver de forma eficiente los problemas de sobrecarga de información, desorganización y falta de accesibilidad en los sitios web. Después, el Capítulo 5 muestra

una arquitectura multiagente genérica para sistemas conversacionales que permite acceder a fuentes de datos heterogéneas de forma simultánea, que son el tipo de problemas que se suelen encontrar hoy en día en los que la información está dispersa por diferentes orígenes, cada uno con una estructura y un formato particular. Posteriormente, los Capítulos 6 y 7 (inglés y español) incluyen una serie de conclusiones y muestran algunas líneas de trabajo futuro. Finalmente, el Capítulo 8 muestra una lista de publicaciones que reflejan el trabajo realizado en esta tesis.

2.6 Estancias de investigación

Parte del trabajo incluido en esta memoria ha sido desarrollado durante una estancia de cuatro meses de duración (1 de Junio de 2009 – 30 de Septiembre de 2009) en el Centre of Computational Intelligence de la universidad De Montfort University de Leicester (Inglaterra), financiada por una beca de Formación de Profesorado Universitario (FPU) del Ministerio de Educación y Ciencia, y bajo la supervisión del Dr. Francisco Chiclana.

This page intentionally left blank

3

Emotions in embodied conversational agents

This chapter presents an innovatory system for controlling the emotional state of an embodied conversational agent. Unlike other existing models, our system is fast adaptable to different application domains and it is highly interpretable. This will provide specialists with a tool to easily test their hypotheses about the most suitable emotional attitude for an agent in a specific domain. It also provides emotional stability and dynamic and automatic behavior orientation, which will result in the design of more believable conversational agents that behave as humans do.

3.1 Introduction

Nowadays there is more and more interest in the development of intelligent systems that simulate the behavior of human beings. One of the most active research fields is made up of virtual conversational agents whose main objective is the de-

velopment of agents which can interact with their environment and behave as people would do. For that reason, agents must behave with a high realism level, so they must be able to emotionally react to events happening in the world they live, seeming to worry about them. This makes evident the necessity of defining an emotional state control system so that agents may be able to behave in a consistent way and adapt themselves to many different situations. Therefore, including a representation and control mechanism of a conversational agent's emotional state is very important.

The work we present in this chapter was included into the research projects "*Intelligent systems for the development of software virtual patients. TIN2007-67984-C02-01. 10/2007-09/2010*" of the Spanish Ministry of Science and Technology, and "*Virtual patient for training and e-training in medicine. P06-TIC-1424. 04/2007-03/2010*" of the Ministry of Economy, Innovation and Science of the regional Andalusian Government. They aimed to develop a virtual conversational agent. In particular, a Virtual Simulated Patient (VSP) (Castro et al., 2010b; López Salazar et al., 2012), a tool for training primary health care medical students. They were carried out by the *Computational Intelligence* [1] research group of the *Department of Computer Science and Artificial Intelligence (DECSAI)* [2] of the *University of Granada* [3], the *IAVANTE Foundation* [4], which provided all the medical support for the development of the VSP, and the *CITIC Foundation* [5]. In addition, they formed the basis for the European project "*MVSPP - Multilingual Virtual Simulated Patient Project [143423-LLP-2008-ES-KA3-KA3MP]*" of the "*Education, Audiovisual and Culture Executive Agency (EACEA). Lifelong Learning Programme. Leonardo da Vinci. Transversal Programme: KA3-ICT.*", which aimed to adapt the Spanish version of the Virtual Simulated Patient to the cultural differences of the native and non-native (minority immigrant population) speakers of seven countries of the European Union: Bulgaria, Germany, Hungary, Italy, Portugal, Spain, and The United Kingdom. The people involved in this project were health care trainers, course

[1] http://ic.ugr.es/
[2] http://decsai.ugr.es/
[3] http://www.ugr.es/
[4] http://www.iavante.es/
[5] http://www.citic.es/

designers, linguistic experts, programming and artificial intelligence computing experts, anthropologists and e-learning designers.

Our problem arose from the way the sanitary training process was carried out. A person was usually trained to play the role of a simulated patient in the presence of students, who were the sanitary professionals to be trained. However, this simulating technique that used human actors had some disadvantages. On the one hand, it needed people playing the role of patients. On the other hand, it was an expensive process that had to be repeated for every new actor and sanitary process to simulate.

In this way, the VSP's development would have the same objectives and advantages as real patients, but opening a new world of possibilities because it would not need people playing the role of patients, and it would make possible training professionals on different subjects, fitting to each one's necessities and without taking a risk for patients.

The aim of the work we present in this chapter was the development of an emotional state control system that would determine the way VSPs act, behaving as people would do, getting a high realism level, which was our main objective. Although we will mainly focus on the VSP domain, in general, this component can be applied to any virtual conversational agent.

The rest of this chapter is organized as follows. First, in Section 3.2, we will analyze some existing models for representing and controlling the emotional state of a conversational agent. Besides, we will describe the requirements of our problem, and we will explain why those models do not cover all of them so that we will define a new system to include some innovatory features. In Section 3.3 we will define the emotional state and personality of an agent, key concepts in the development of an emotional state control system. After that, in Section 3.4, we will analyze in depth the fuzzy rule-based systems that make the transitions between emotional states. Next, in Section 3.5 we will analyze the advantages of introducing a probabilistic component and a dynamic evolution mechanism into the system. In Section 3.6 we will carry out several tests in order to measure the accuracy of the system. Finally, we will make a conclusion and we will show the future work in Section 3.7.

3.2 Emotional state control models

In this section we will analyze some emotional state control models in order to see what features they present and know if they satisfy all the requirements of our problem. In Section 3.2.1 we will study different approaches for the internal representation of the emotional state of a conversational agent. Next, in Section 3.2.2 we will analyze the structure of some of these models. Finally, in Section 3.2.3 we will mention several emotional state transition mechanisms. Based on these models and the requirements of our problem, we will define a set of features to be considered for the design of the system. All these elements will be discussed in Section 3.2.4, where we will also pose the necessity of defining some improvements and novelties regarding existing models to cover all the requirements of our problem.

3.2.1 Representation models

There are several models for representing the emotional state of a conversational agent. In this section we are going to analyze two approaches, the basic emotions model of Yanaru et al. (1994) that follows the idea presented by Ekman (1984) in his basic emotions theory, which asserts that there is a set of basic emotions common for every culture, and the circumplex model of affect described by Russell and Feldman Barrett (1999).

The basic emotions model proposes an n-dimensional representation based on some postulates from Plutchik (1960):

- there is a small number of pure or primary emotions,
- the rest are mixed emotions, that is, they can be obtained combining primary emotions,
- and each emotion can exist at different intensity levels.

It also asserts that emotional state transitions depend on three factors:

- sign: activities can be approximative (positive) or evasive (negative).

- intensity: the intensity of the input signal affects the emotional process.
- length: emotions vary depending on how long the excitement is.

The number of emotional attributes can vary a lot among publications, although authors usually choose eight emotional attributes, so that emotional states are vectors in that eight-dimensional space. The choice of these attributes also depends a lot on the specific domain. Yanaru et al. (1994) define the following emotional attributes: joy, sadness, anger, fear, expectation, surprise, hatred and acceptance. Based on these attributes, these authors follow the approach by Plutchik (1960) that suggests 68 emotional states, 34 primary and 34 secondary. However, this amount of emotional states is quite excessive in practice since it is quite difficult to distinguish between many of them. For every vector, emotional attribute components belong to [-1,1], although using [0,1] is also frequent.

Russell and Feldman Barrett (1999) present an alternative to this model. According to these authors, the basic emotions model is based on a sequence of discrete and mutually excluding basic categories, meaning that if an agent feels two emotions simultaneously, each emotion belongs to one and only one basic category. They also assert that this discrete representation is poor regarding the continuous nature of the valence and arousal dimensions of the circumplex model. Furthermore, they say that categories are not mutually excluding but fuzzy, since they cannot be divided into crisp values, as it is shown in Figure 3.1.

Therefore, in the circumplex model of affect, emotion transitions are easier to represent visually since we could gradually change from one to another varying some facial parameters of the agent's face, whereas with the basic emotions model we would have to specify these transitions more in detail. However, the inclusion of new emotion types in this model is much more complicated than in the basic emotions model. This is a serious problem since different application domains may require different emotion types, so we need to be able to easily add new emotions. For example, in our application, the patient must be able to show an embarrassment emotional state, which does not appear in this circumplex model and the addition of this emotion into this model is quite complicated, if possible.

Based on these two models, for representing the emotional state of our conversational agent we have chosen the basic emotions model since it allows repre-

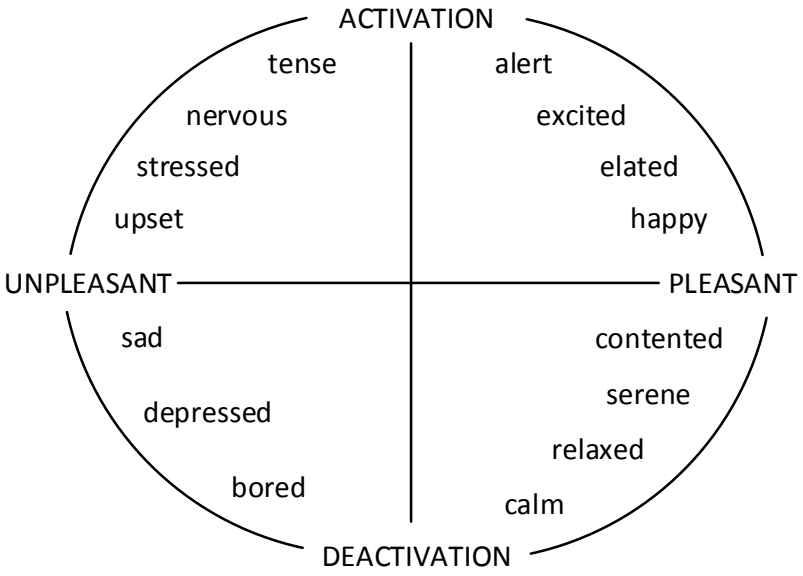


Figure 3.1: Circumplex model of affect

senting each emotional attribute independently. In this way, new basic emotions can be included easily and quickly if the specific domain requires it. Moreover, this model allows the appearance of more than one basic emotion simultaneously, whereas the circumplex model of affect would not allow this if these emotions are not correlative. For example, suppose that an agent is worried about his bad health and also angry because somebody is asking him the same question several times. As these basic emotions are independent, if the subject of the conversation changes, the agent could stop being angry, but he would keep his worry level about his health.

3.2.2 Structures

Nowadays there are several models for controlling the emotional state of a conversational agent. Some of them are based on the OCC (Ortony, Clore and Collins) theory of emotions (Ortony et al., 1988), which comes from psychology. This theory considers that emotions are valence responses, that is, good or bad, of agents to events, taking their own goals and motivations into account. Other works

(Rosis et al., 2003) keep away from this approach based on goals, standards and preferences, and use other structures in order to get virtual agents that behave affectively as humans do. Apart from the general scheme of the emotional state control system, the different approaches vary in the components they take into account in the design of the system. For example, some works represent emotions in a discrete way, others do not take the agent's personality into account or the fact that emotions must decay over time, and so on.

As Bartneck (2002) says, the OCC model distinguishes between 22 emotional categories, which is too complex for the development of a believable conversational agent. He states that this model was originally created thinking of structuring and controlling human emotions, so it may be too complex for its application to the conversational agent's domain. It would be reasonable to simplify the number of emotions, so that only the emotional categories that the agent will really use were handled. In addition, in this model some emotions are highly related, like gratification and gratitude, which only differ about the person who carries out the action. Moreover, if the agent is not going to have a user model, he will not be able to evaluate the consequences that events have on him, so this part should be excluded from the emotional state control system. A serious problem with this model (for our purposes) is that, in spite of representing a great number of emotional categories, some emotions are not included and they do not match any other, like surprise. Therefore, our system should not only include the emotional categories that our problem requires, but it must also be adaptable, allowing easily including new emotion types if necessary, which is quite complicated in the OCC model.

An interesting approach based on the OCC theory of emotions is *FLAME* (*Fuzzy Logic Adaptive Model of Emotions*) (El-Nasr et al., 2000). It is a fuzzy system which transforms the impacts that events have on the agent's goals into emotional intensities. It includes a mechanism for learning relations between objects, event sequences and user expectations, which allows the agent to dynamically adapt his responses. In addition, they state that emotions decay over time, and they establish an emotional filter. However, in spite of the fact that many features of this model are really interesting, it does not include the agent's personality, which is essential to cover all the requirements of our problem and

get realistic conversational agents.

Rosis et al. (2003) present a conversational agent called Greta, which uses dynamic belief networks to give eating advice to users. They use a subset of the emotions described by the OCC theory. They take account of personality, emotions overlapping and emotion decay over time according to the agent's personality and context. They also weigh the agent's goals in order to know when to show or hide emotions. However, this model involves some difficulty when calibrating the parameters of the networks.

Cathexis (Velásquez, 1997) employs a distributed architecture. It includes several sensors for detecting internal and external stimulus that vary the intensity of emotions. This model also considers that emotions decay over time, although it does not include any adaptive mechanism and it does not take the agent's personality into account.

Prendinger and Ishizuka (2001) propose an agents architecture for modeling and filtering emotions. They think that, in addition to be based on emotions and personalities, agents must take their social context into account. For example, if a sales agent gets angry, according to his social role, he can show his colleagues this emotion, but he must be conscious of not being angry with his customers, so that he will try to hide his emotional state. Although using an emotional filter to distinguish the emotional state from the emotion shown is a good idea, this model has a weak point. They assign a set of five discrete intensities to every emotional state and personality, which is not enough to model them in a smooth way. Another interesting consequence of this work is that agents may act in groups, capturing the emotional state of another agent and pretending to feel his emotion even though their actual states were completely different. This idea is similar to that proposed by Elliott (1992) in his *Affective Reasoner*, a taxi drivers' world in which they can feel 24 different emotions in response to other drivers and events happening in the world. These agents have personality and react differently in different situations. Some of them can get angry with their passengers when they get small tips, whereas others can consider that this is part of their job. Others can hide these negative emotions or even pretend positive emotions instead. However, this model considers that emotions have no intensity.

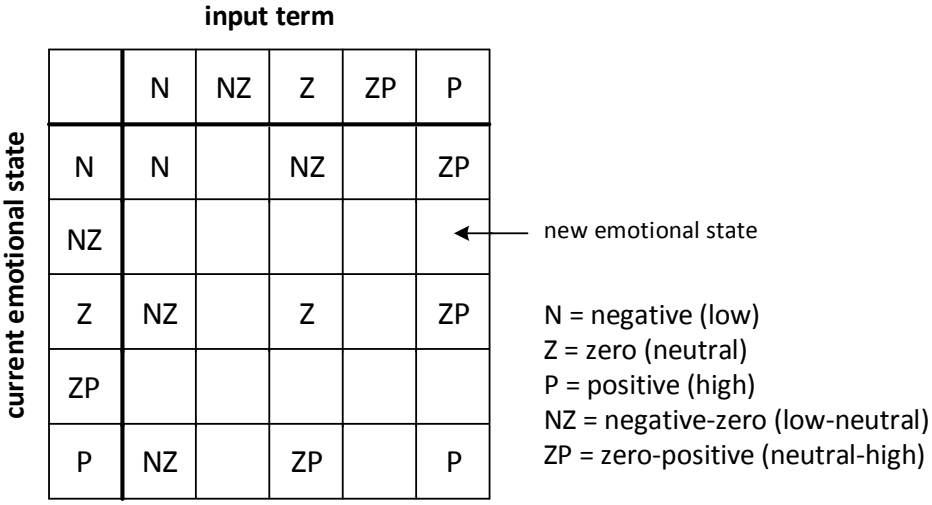


Figure 3.2: Yanaru's fuzzy inference mechanism

3.2.3 Transition mechanisms

Apart from the structure of the emotional state control system, it is necessary to define an updating mechanism for the agent's emotional state. Yanaru et al. (1994) make transitions between emotional states by means of a fuzzy inference mechanism using a sequence of terms that belong to the words of the text as input. Every word suggests several emotional states, between the 68 proposed by Plutchik (1960), that are combined to get only one emotional state, which will represent that word. In this way, the new emotional state is inferred from the agent's current emotional state and the word or input term using a fuzzy system with nine rules as it is shown in Figure 3.2. This fuzzy inference process, which is performed individually for every emotional attribute, is based on a triangular membership function. However, although this transition mechanism is quite interesting, it does not take account of very important features like the agent's personality.

This updating model is similar to that presented by El-Nasr et al. (2000) in *FLAME* in the sense that it uses a fuzzy rule-based system. However, in this case, since it is based on the OCC model, rules do not calculate emotional

state variations straight away, but they are used to get desire and expectation levels, which intensity value for every emotion is established from, using a set of formulas.

Another model, proposed by Egges et al. (2003), uses a lineal approach with two functions for updating the agent's emotional state, which is represented by vectors. The first function depends on the agent's personality as well as a previous emotional state record, whereas the second represents an internal change that decreases the emotional state. Although this model could behave reasonably well in some cases, we consider that a lineal system is not enough to create realistic virtual agents with a behavior that may adapt to very different situations.

A very important question that we will have to take into account when calculating the emotional state transition is the decay function, which will make emotions decay over time. As we have seen, many models make this consideration. We believe that the decay function must depend on the agent's personality (each one will tend toward a different equilibrium according to his personality type) and his emotional state. This question will be analyzed more in depth in Section 3.5.

3.2.4 Components

Now we are going to analyze all the requirements of our problem. Some of them will be components and features that other authors have taken into account for the development of emotional state control systems in embodied conversational agents. Besides, it will be noted the necessity of defining some improvements and novelties to cover all these requirements. In this way, a new system will be developed to solve our particular problem.

Realism This is the most important feature that our system must have. Agents must react in an emotional way to the events happening in the world they live. In this way they are more believable than if they remain impassive in the presence of such events. In addition, agents must act as naturally as humans do.

Gradualness Emotions are dynamic and happen at different intensity levels. Therefore, an agent must react differently if he is a little sad or very sad.

Personality It is a static factor with a double function. First, it makes agents different, and second, it influences the way agents capture the world and behave.

Emotional filtering Agents must be able to hide and pretend emotions according to their social context and personalities.

Temporal decay Emotions do not stand forever. They tend to reach an equilibrium when no events happen.

Reuse The system must be as flexible and domain independent as possible so that it may be reusable.

Adaptability The system must also be able to capture specific domain knowledge. For that reason, it must allow some adaptability, making it possible, for example, to change the chosen emotional attribute set.

Interpretability It must be easy to understand, even for not specialized people, so that experts at different application domains can help us to capture the specific knowledge. In this way we can get more believable agents. Hence, it cannot be like a black box, but we need to understand why an agent behaves in a particular way.

Non-determinism Agents' behavior must be variable, even when facing the same situation at different times.

Stability Agents' behavior must also have some stability, that is, emotions cannot appear and disappear too quickly.

Memory It is necessary for agents to follow the course of the conversation so that they may take account of, for example, if somebody asks them the same question again and again to act accordingly.

Conduct types One of the features that we consider more relevant to get realistic conversational agents is that the system must lead automatically and dynamically the agent's behavior.

Agents' interaction Agents must be able to hide and pretend emotions according to other agents' behavior.

Agent's Health In our problem, agent's health needs a specific and exhaustive control.

As analyzed models did not cover all the requirements of the problem we had to solve, it was necessary to develop a new system. This system would have some features that already existed in those models like agents' personality, as well as some improvements and innovatory elements such as fast adaptability to any application domain of conversational agents (not only teaching or training), good interpretability (which will allow specialists to help us to capture specific knowledge), emotional stability, dynamic and automatic agents' behavior orientation, and so on. All of this would result in the design of more believable conversational agents.

To sum up, although there were many emotional state control systems, non-covered or improvable necessities made evident the importance of developing a new system that would be able to solve our problem with total accuracy. This system will be described along next sections.

3.3 Representing emotional states and personalities

The emotional state control system of an embodied conversational agent is based on two essential concepts like emotional state and personality, which we are going to describe along this section.

Following the approach by Ekman (1984) and Yanaru et al. (1994), the emotional state of a conversational agent will be built on the basis of a set of basic

Emotional State	Emotional Attributes							
	Joy	Dis.	Ang.	Fear	Wor.	Sur.	Sad.	Emb.
State 1	0.73	0.26	0.12	0.08	0.11	0.39	0.04	0.10
State 2	0.05	0.03	0.07	0.48	0.39	0.10	0.86	0.01

Legend: Dis(dain); Ang(er); Wor(ry); Surp(rise); Sad(ness); Emb(arrassment);

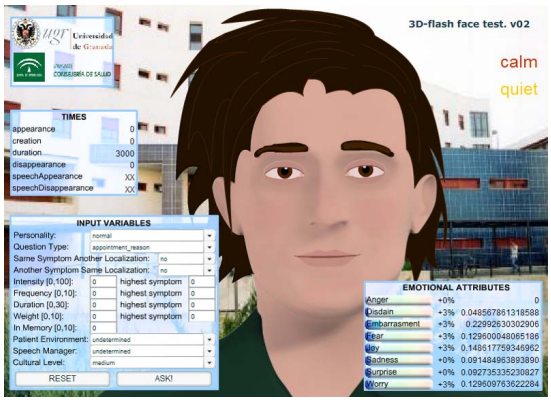
Table 3.1: Some emotional states

emotions which we are going to call *emotional attributes*. Every attribute, eight in our system, corresponds to one dimension of the space in which emotional state vectors will be represented. These eight dimensions correspond with the following emotional attributes: joy, disdain, anger, fear, worry, surprise, sadness and embarrassment. In other words, we can say that emotional attributes are to emotional states the same as *Red*, *Green* and *Blue* components are to colors in the *RGB* color model. The *IAVANTE Foundation* staff have helped us to choose these emotional attributes. Although this set allows covering more or less the main emotions an agent can feel, our system allows modifying it in a fast and easy way in case the application domain requires it. The value for every attribute will be a real number between 0 (total absence) and 1 (total presence). Table 3.1 shows two examples of emotional states, which can belong to different agents or to the same one but in different times.

As can be seen in Figure 3.3, the graphical representation of the conversational agent was initially made by means of a 2D model which simplified a little all the emotion management process. It was a Flash [1] component developed from scratch and designed specifically for web browsers. It used 2D animation techniques and the face of the agent was divided into several regions that were handled independently to give a feeling of depth. However, it was not able to show several emotions at the same time, and only the one with the higher intensity value was selected.

Later, this initial 2D model was replaced by a more advanced 3D model which allows a higher degree of realism (especially when using real people photographies), the combination of basic emotions, and improves the transition between different types of emotions. As Figure 3.4 shows, this new version is made up of

[1] http://www.adobe.com/products/flash/



(a) Normal (cartoon, 2D model)

(b) Normal (real person, 3D model)

(c) Joy (cartoon, 2D model)

(d) Joy (real person, 3D model)

(e) Disdain (cartoon, 2D model)

(f) Disdain (real person, 3D model)

Figure 3.3: Emotional states (normal, joy, and disdain)

(g) Anger (cartoon, 2D model)

(h) Anger (real person, 3D model)

(i) Fear (cartoon, 2D model)

(j) Fear (real person, 3D model)

(k) Worry (cartoon, 2D model)

(l) Worry (real person, 3D model)

Figure 3.3: Emotional states (anger, fear, and worry)

(m) Surprise (cartoon, 2D model)

(n) Surprise (real person, 3D model)

(o) Sadness (cartoon, 2D model)

(p) Sadness (real person, 3D model)

(q) Embarrassment (cartoon, 2D model)

(r) Embarrassment (real person, 3D model)

Figure 3.3: Emotional states (surprise, sadness, and embarrassment)

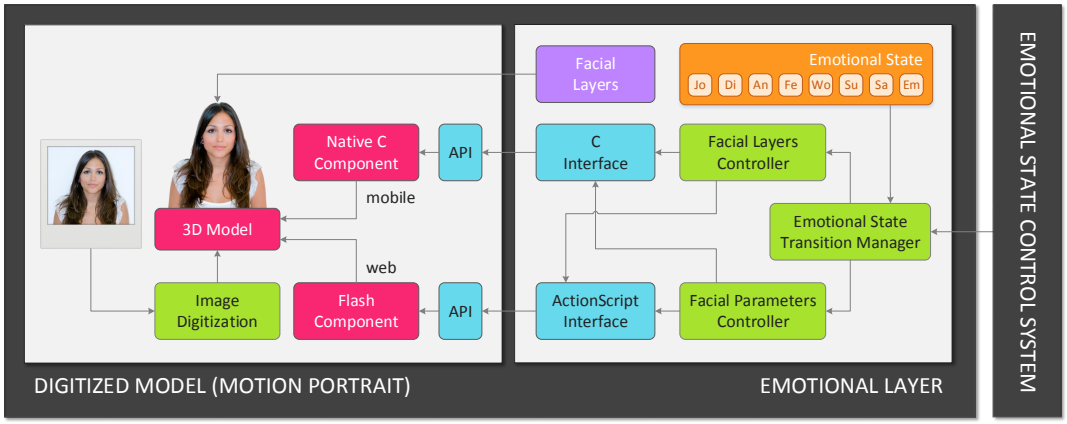


Figure 3.4: Components of the 3D model

a digitized model and an emotional layer. The 3D model is created from a single photography of a real person or a cartoon using the technology of Motion Portrait [1]. It is available as a native C component for smartphones, or a Flash component for web browsers, and it can be controlled by means of an API (Application Programming Interface). On top of this model, we have made an additional layer which defines emotional attributes and provides emotion management. This layer includes a system for controlling the facial parameters of the avatar. Moreover, it defines new facial layers and effects that are necessary for the different emotions (e.g. becoming embarrassed).

On the other hand, personality is a set of static features, associated with each agent, that make him tend to his equilibrium, which will vary according to his personality. Besides, it will allow agents to carry out a subjective assessment of the context of the world around them. An agent's personality will be defined, as his emotional state, on the basis of the eight emotional attributes described before. Personality types, which can be seen in Table 3.2, have been determined with the help of the *IAVANTE Foundation*'s specialists. These types cover, a priori, all personalities that are supposed to be adequate for a conversational agent, although the designed emotional state control system is independent of the defined personality types, so that it allows adding or removing new types or

[1] http://www.motionportrait.com/

Personality	Emotional Attributes							
	Joy	Dis.	Ang.	Fear	Wor.	Sur.	Sad.	Emb.
Anguished	0.00	0.00	0.00	0.60	0.60	0.00	0.00	0.00
Depressive	0.00	0.00	0.00	0.00	0.30	0.00	0.60	0.40
Hypochondriac	0.00	0.00	0.50	0.70	0.00	0.00	0.00	0.00
Maniac	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Phobic	0.00	0.00	0.00	0.60	0.30	0.00	0.00	0.00
Normal	0.10	0.00	0.00	0.10	0.10	0.10	0.10	0.20

Legend: Dis(dain); Ang(er); Wor(ry); Surp(rise); Sad(ness); Emb(arrassment);

Table 3.2: Personality types

even modifying the existing ones if necessary, without posing a problem to the system.

When assigning a particular agent's personality, a little variation in the value of the emotional attributes associated with that personality is carried out in order to achieve some randomness. In this way, two agents that share the same personality do not have to be exactly the same.

In the next section, we will describe the structure of our emotional state control system, once we have defined the two essential concepts which it is based on, emotional state and personality.

3.4 Building the fuzzy rule-based system

In Section 3.2 we reviewed some models for controlling the emotional state of a conversational agent. Some of them were based on the OCC theory of emotions which represent emotions as agents' responses to events, taking into account their objectives and preferences, and a sequence of standards. Other models carried out an emotion representation based on a set of basic categories. To update every emotional attribute independently and in parallel, we have chosen a set of fuzzy rule-based systems the features of which will be described along this section and the next one. We have chosen this kind of system because:

- it allows a natural representation,
- the knowledge can be added and modified easily,

- it can be applied to a wide range of different domains without mattering the complexity level,
- it allows us to vary the set of emotional attributes (or basic emotions) easily, so that we only have to create a new attribute and specify the rules that modify it,
- it is reusable because there will be a set of domain independent rules that can be applied to any domain, as well as another set of domain dependent rules specific for each application domain that will help us to create realistic conversational agents,
- and it is independent for each emotional attribute.

Along this section we will describe all the components of these fuzzy rule-based systems.

3.4.1 Inputs of the fuzzy rule-based systems

As we have explained before, one of the problems that may arise when we use a fuzzy rule-based system is the fact that we have to create a new set of rules because the specific information of the concrete domain, and therefore the input variables of the system, have changed. To prevent this as far as possible there is a set of input variables, shared by all applications, that determines a set of rules that are suitable for all domains. Moreover, for every specific domain it will be necessary another set of input variables which may capture that specific knowledge and whose rules will only be suitable for a certain domain.

In general, the emotional state of any conversational agent, independently of the application domain, will depend on the following input variables:

Personality Any of the previously defined personality types, although they might vary slightly from an application to another if we wanted to capture a more specific knowledge. As we have mentioned before, personality will determine the way agents perceive all the events that happen in the world and how their actions are reflected through emotions.

Health The emotional state of an agent is going to be influenced by his health, as it happens to people. For example, when we are ill or tired, we are more prone to get angry with others. Therefore, agents must act in a similar way.

Environment This variable will control the agent's interaction with all the individuals of the world (other agents or even people) that might have influence on his emotional state. Although this is a common variable for many applications, its content may vary from one domain to another because the agent's situation may be very different.

Conversation type Although the kind of conversation to have with the agent depends on the specific domain, this variable must appear in all the fuzzy rule-based systems independently of the application domain because this will be the way we will communicate with the agent. Even, there will be some common conversation types like positive or negative comments, which can vary the levels of the agent's attributes *joy* or *angry*.

Memory How many times each question is repeated [0,10]: one of the most important components that an agent must have to be believable is the conversational record. It would be useless that the agent reacted perfectly when he faced a stimulus and showed an emotion according to such reaction if after repeating the same stimulus again and again the response were always the same. For that reason, it is necessary to take the conversational record into account so that, for example, the agent can get angry if he thinks that somebody is pulling his leg because he is repeating the same question again and again.

Speech manager If the structure of the dialog has been prefixed, the system must control the conversational flow.

Once the generic input variables of the system have been defined, we have to establish the specific variables that are going to capture the domain specific knowledge. For example, if we consider the VSP application, we must include

the following set of variables so that agents can act with more naturalness in this specific domain:

Conversation type Since in this domain the objective of the conversation is to determine the patient's illness, and the doctor will ask him some questions, from here on we will refer to this variable as `QuestionType`. The question types that the doctor asks the patient are related to the appointment reason, a particular symptom, sexual content questions, positives or negatives comments, and silence (keeping quiet). To incorporate a new question type to the system we only have to assign it a linguistic label and create the associated rule that will store that knowledge.

Same Symptom Another Location, Another Symptom Same Location {yes, no}: the system must establish relations between symptoms and body locations. For example, if the doctor asks the patient about a particular symptom at a certain location, and the patient suffers from that symptom but at another location, he must be able to relate both concepts and react in a similar way that if the doctor finds both the symptom and the location.

Intensity [0,100] of the symptom the doctor asks about.

Frequency [0,10] Times a day the VSP suffers from the symptom the doctor asks about.

Length [0,30] Number of days suffering from the symptom the doctor asks about.

Weight [0,10] Importance of the symptom the doctor asks about (0 is none, 10 is maximum).

Health In this application it is necessary a greater detail level because it belongs to the medicine field. The system must take account of the highest intensity, frequency, length and weight values of all the symptoms the patient suffers from.

Environment Aspects of the patient's environment that can influence his emotional state such as if a relative suffers or has suffered from a particular disease. Therefore, if for example the doctor asks the patient about a particular symptom that has caused the death of a relative of him, the patient will tend to increase the levels of his attributes *fear* and *worry*.

Speech manager The speech manager controls the flow of the doctor-patient conversation on the basis of a set of states: introduction, information explanation, illness diagnosis, farewell and silence. Transitions between states will be useful for updating the patient's emotional state. This will allow the patient, among other situations, to get angry if the doctor makes a diagnosis as soon as he sees the patient getting into the surgery, or if the doctor says goodbye without making a diagnosis.

Cultural level {low, medium, high}: although this parameter is the one that has less influence in the patient's behavior, a patient with a high cultural level could have a greater knowledge of the possible illness that he is suffering from, so the levels of his attributes *fear* and *worry* may increase more regarding a patient with a lower cultural level.

3.4.2 Output of the fuzzy rule-based systems

The fuzzy systems will be Mamdani so that a greater interpretability will be obtained in the rules that define the variation level of every emotional attribute. Therefore, the consequent of every rule will be determined by a linguistic expression that will contain the specific variation level to produce in every emotional attribute, instead of employing a formula. This variation will be defined through one of the following eight labels: high negative, medium negative, low negative, zero, low positive, medium positive, high positive, and personality. Personality label is worth an especial mention. Rules that present this kind of variation will be dynamic rules, that is, they will dynamically adjust the increase or decrease level of the emotional attributes according to the agent's personality and his current emotional state. In Section 3.5 we will describe how this kind of dynamic

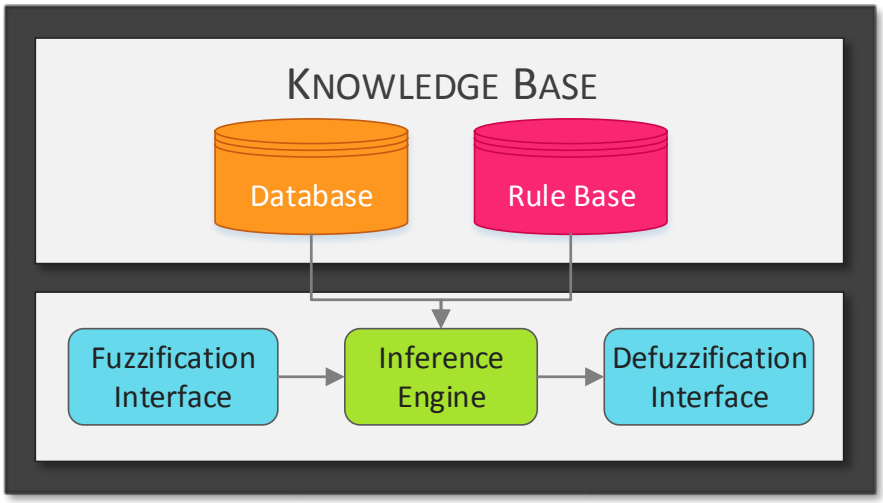


Figure 3.5: Structure of the fuzzy rule-based systems

rules works. Once the outputs of the fuzzy rules have been obtained, they are aggregated to yield a single variation level for each emotional attribute.

3.4.3 The fuzzy rule-based systems

The fuzzy rule-based systems that will control each emotional attribute of the agent will have three components, as can be seen in Figure 3.5: the knowledge base (which contains the database with the linguistic variables and labels, and the rule base), the inference engine (which estimates the output value according to the input values), and the fuzzification and defuzzification interfaces (which respectively convert crisp values into fuzzy ones, and vice versa).

As we have said, the knowledge base is made up of a database that contains a set of linguistic variables. These variables are related to the inputs and the output defined in Sections 3.4.1 and 3.4.2, respectively. For the numeric linguistic variables, such as intensity or frequency, a numeric membership function must be defined. For the choice of the membership function type we must consider that on the one hand we need that it may allow high degrees of membership (almost one) for a range of input values (an interval), so triangular membership function is refused, and on the other hand we do not want an excessively powerful function,

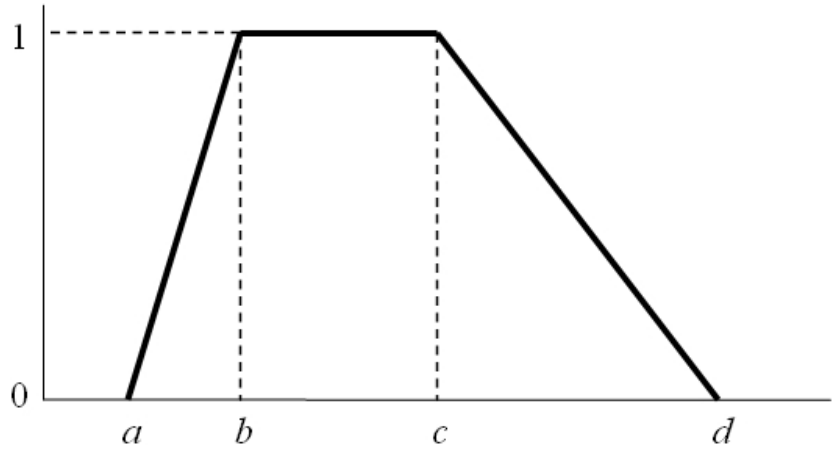


Figure 3.6: Parameters of numeric membership functions

since this would suppose doing more operations when computing the degree of membership of functions, so that the global performance would decrease, and we need a real time system. Therefore, we will use trapezoidal membership functions, which will allow us to model the values of the linguistic labels perfectly. To define a trapezoidal membership function the values of four parameters a, b, c, and d must be set, corresponding to the corners of the trapezium, as can be seen in Figure 3.6. Note that experts have helped us to set the membership functions of all the variables of the system.

From these linguistic variables and labels, the fuzzy rules of the rule base are defined. In this way, a fuzzy rule will have two components, the antecedent (left side of the rule) and the consequent (right side), although as we will see further on, the antecedent will be an optional part. Therefore, a fuzzy rule will be defined by means of an expression similar to the following

$$[V_1 : [!]L_1 \quad V_2 : [!]L_2 \quad ... \quad V_i : [!]L_i] \quad \Rightarrow \quad V_o : L_o$$

where V_1, V_2, ... V_i are input linguistic variables, L_1, L_2, ... L_i are any of the labels allowed for these variables, V_o is the output variable, and L_o is the label of this output variable (the variation level of the emotional attribute). Optionally, the labels of the input variables can appear negated at the antecedent of the rule. Moreover, the antecedent is an optional part, so a rule can be formed by zero or more fuzzy expressions $V_i : [!]L_i$.

```
Personality:!Depressive
∧  QuestionType:DoctorPositiveComment
∧  InMemory:Zero
⇒  Variation:LowPositive
```

Figure 3.7: Example of rule for the emotional attribute *joy*

```
QuestionType:Symptom
∧  Intensity:High
⇒  Variation:LowPositive
```

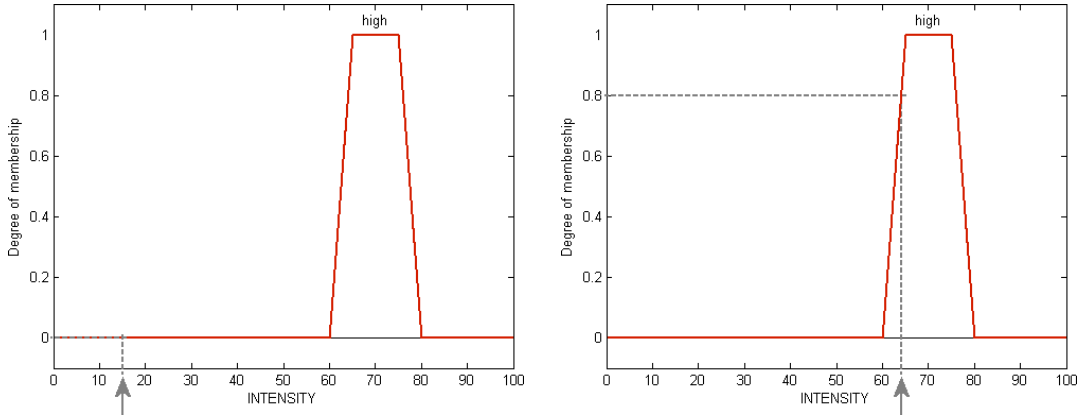
Figure 3.8: Example of rule for the emotional attribute *fear*

Figure 3.7 shows an example of rule for the emotional attribute *joy*. This rule is interpreted as follows: IF the patient does *not* have a *depressive personality*, AND the *question type* that the doctor asks is a *positive comment*, AND this is the *first time* the doctor makes a positive comment, THEN a *low positive variation* of the corresponding emotional attribute is produced (*joy* in this case).

In addition, every rule has a probability value associated with it that corresponds to the probability that the rule has to be taken into account at the current state of the system to calculate the emotional state transition. The rules and the probabilities associated with them have been determined with the help of specialists based on the common sense and studies about people's behavioral tendencies. In Section 3.5 we will analyze this feature of the system in depth.

To know if a rule is fired or not, the values associated with every input variable of the rule are first assigned. Next, every fuzzy expression of the antecedent is evaluated. For example, consider the rule of the emotional attribute *fear* that appears in Figure 3.8. This rule points out that IF the doctor asks the patient about a certain *symptom*, AND the patient suffers from it with a *high intensity*, THEN his fear level will be *lowly increased*.

As we have said before, to evaluate a rule the first step is to assign the values of the input variables. For this example, we will only show the values of the two variables that affect the rule. Assume that the degree of membership of the fuzzy expression `QuestionType:Symptom` is 1, that is, the doctor has asked the patient about some symptom (note that if the `QuestionType` had been a `PositiveComment` or a `Diagnosis`, for example, the degree of membership of this



(a) Intensity value: 15 (b) Intensity value: 64

Figure 3.9: Degree of membership to the fuzzy expression `Intensity:High`

fuzzy expression would have been 0). Moreover, suppose that the patient has an intensity equals 15 for the symptom the doctor has asked about, so that the degree of membership of this value to the fuzzy expression `Intensity:High` is 0, as can be seen in Figure 3.9a. Therefore, the rule would not be fired in this case.

Now suppose that the intensity value of that symptom is 64, so the degree of membership is 0.8, as Figure 3.9b shows. The rule would be fired in this case since the minimum degree of membership is now 0.8.

The next step is to calculate the output value, that is, the emotional attribute variation. The outputs of all fired rules must be aggregated. To perform this aggregation we use the *FATI* approach (*First Aggregate, Then Inference*), so that all output fuzzy sets are aggregated to get a concrete emotional attribute variation. In addition, the probabilistic component of the system takes into account the degree of matching of every rule to the input values to consider or not the associated output fuzzy set in the aggregation step. We will talk about this in Section 3.5.

To determine which variation best represents such fuzzy set aggregation, we use the center of sums method to consider the contribution of each area independently, adding all fuzzy sets instead of joining them. So, if any area is repeated, it is considered again, reinforcing in that way overlapped zones. Therefore, if only the rule of Figure 3.8 was fired, the aggregation would yield an increment at the value of the emotional attribute *fear* equals 0.09, which is the

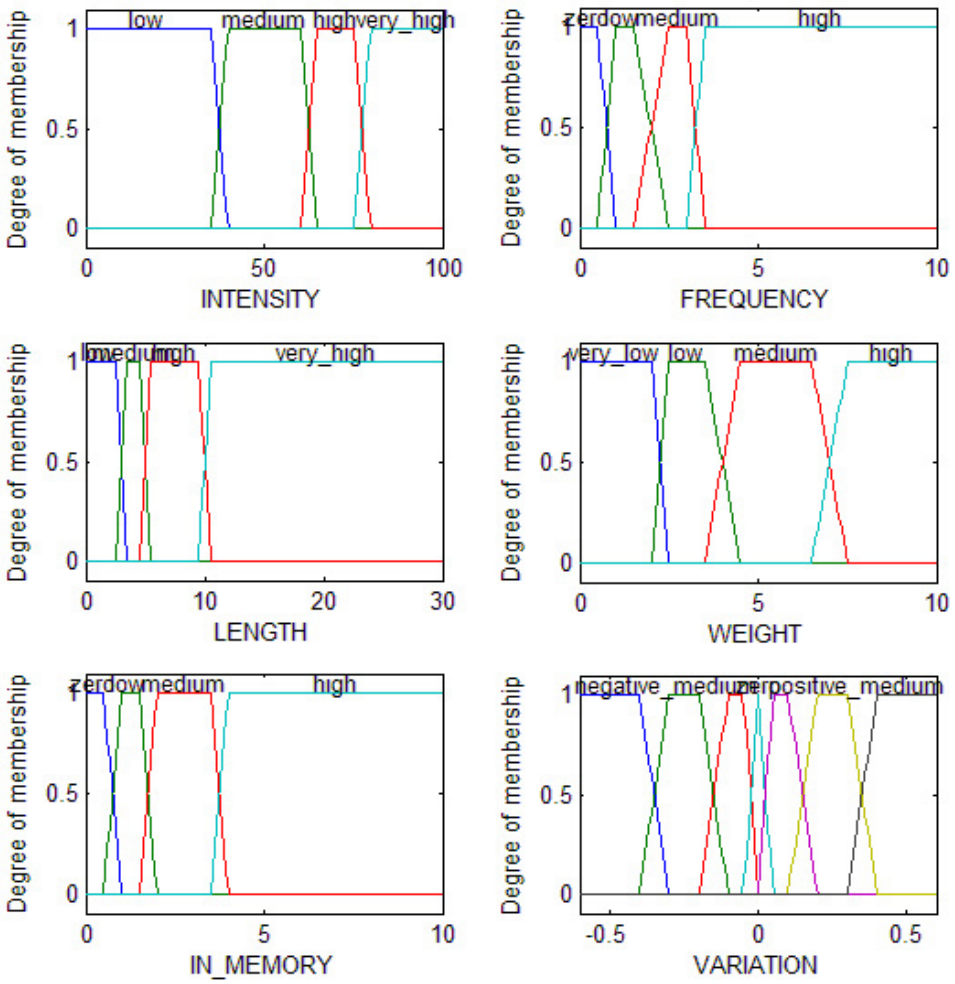


Figure 3.10: Numeric linguistic variables and their corresponding labels

value of the center of sums for the part of the trapezium of the fuzzy expression `Variation:LowPositive` under value min{1, 0.8} = 0.8 (Figure 3.10 shows the linguistic labels of all the numeric linguistic variables of the system).

With regard to the fuzzy rule types of the system, we can identify the following:

- rules that do not vary the emotional attributes.
- rules that vary the emotional attributes according to the agent's personality.
- rules that increase the emotional attribute values.

- rules that decrease the emotional attribute values.

The first two kinds of rules are worth a special mention. In order to get some stability in the emotional states of the conversational agent, the fuzzy rule-based systems include several rules the output of which is an emotional attribute null variation. In this way, these rules, when fired, decrease the impact that the rest of fired rules have in the agent's emotional state.

On the other hand, there is another interesting and needed set of rules that produces as output an emotional attribute variation that depends both on the agent's personality type and his current emotional state. This kind of rules are dynamic, since their specific variation levels change through the different iterations of the system. In Section 3.5.2 we will see how this kind of rules work.

3.5 Dynamic evolution of the fuzzy rule-based systems

Two of the most novel and important elements of our emotional state control system are its probabilistic component and its dynamic evolution mechanism. In general, the probabilistic component will make our system non-deterministic, which will result in more natural and different conversational agents because neither all agents will be the same nor the same agent will always behave in the same manner when facing the same situation. On the other hand, the dynamic evolution mechanism of the control system will allow us to adjust, in runtime, both the probabilities associated with every rule and the variation levels of the consequents of the dynamic rules. Figure 3.11 shows the general structure of these features of the system. First of all, the set of rules to evaluate in the current iteration of the system is updated according to the probabilities of all the rules (a draw is carried out for this purpose). Next, the selected rules are evaluated for the new input values and the outputs of the fired rules are aggregated to produce only one attribute variation for each emotional attribute. Those variations lead the agent to a new emotional state, which modifies both the consequents of dynamic rules and the agent's behavior. Depending on that new behavior, the probabilities of some rules may be modified. At the moment, the dynamic evolution mechanism

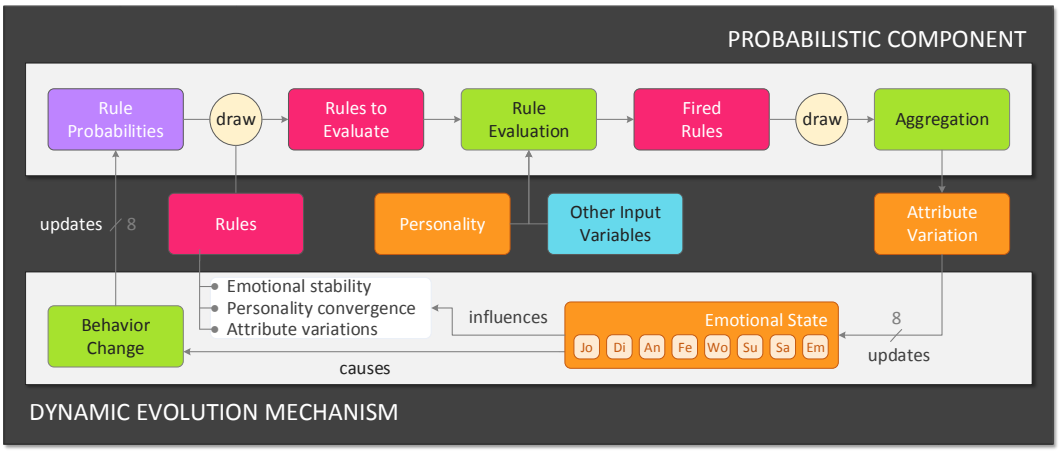


Figure 3.11: General structure of the probabilistic and dynamic components

keeps these probabilities constant because we have not enough information about how the different behavioral patterns evolve in the course of time, according to the human psychology.

3.5.1 The probabilistic component

In Section 3.4.3 we said that every rule of the fuzzy system had a probability associated with it, which was going to modify the set of rules to be taken into account at every state of the system to calculate the next emotional state transition. We also said that the probabilistic component was going to consider the degree of matching of each rule regarding the input values to take or not the corresponding output fuzzy set into account in the aggregation process.

Therefore, with this probabilistic component we want to achieve a triple objective. On the one hand, since we are using probabilities, the fuzzy rule-based system will be non-deterministic because the same input will not always yield the same output. This will allow agents to behave differently even if they share the same type of personality. Besides, the same agent will be able to behave in a different way when facing the same situation at different times in spite of having the same input values. On the other hand, associating a probability with every rule will cause that in every iteration of the system may there be a set of

```
⇒  Variation:Zero (0.20)

Personality:!Depressive
∧  QuestionType:PositiveDoctorComment
∧  InMemory:Zero
⇒  Variation:LowPositive (1.00)

Personality:!Depressive
∧  QuestionType:PositiveDoctorComment
∧  InMemory:High
⇒  Variation:LowPositive (0.50)
```

Figure 3.12: Three rules for the emotional attribute *joy*

rules that will not be evaluated. In this way, we can achieve, in addition to the variability in the agent's response, some stability in his behavior because a more reduced set of rules will be considered to calculate the emotional state variation. Finally, the use of the degree of matching to determine if every fuzzy set must be included or not in the output aggregation process will control that the influence of those sets with a lower degree of matching may be lower.

For example, consider the three rules belonging to the emotional attribute *joy* that are shown in Figure 3.12. The associated probability appears next to each rule. Assume that the agent's personality is non-depressive, for example normal, and suppose that the doctor makes a positive comment to the patient and it is the first time he does since the patient got into the surgery. If we did not consider the probabilistic component, under these conditions the first two rules would always be fired, so that, after aggregating all the outputs, a positive variation under the low value would be yielded, as can be seen in Table 3.3. On the other hand, if we considered the probabilities of the rules, in this case the output would be low positive with a probability equals 0.8 (when the second rule is taken into account but not the first one) or, as if we did not consider probabilities, a little under the low value with a probability equals 0.2 (when both rules are taken into account). In this way, the probabilistic component would not yield in this case too much variation at the output (it would only be slightly increased in the 80 percent of the cases). Therefore, the patient would slightly increase his level of the emotional attribute *joy*.

Probability	Variation	
	without probabilities	with probabilities
0.8	`LowPositive` ↓	`LowPositive`
0.2		`LowPositive` ↓

Table 3.3: Weak influence of the probabilistic component

Probability	Variation	
	without probabilities	with probabilities
0.5	`LowPositive` ↓	`Zero`
0.4		`LowPositive`
0.1		`LowPositive` ↓

Table 3.4: Strong influence of the probabilistic component

Now, assume that the doctor has made several positive comments to the patient so that the expression `InMemory:High` is true with total certainty (degree one). In this case, the first and the third rule would be fired. Without considering probabilities, the response of the system would be, as in the previous case, a positive variation and slightly lower than the low value. However, if we take the probabilistic component into account in this case, the system may yield a quite different behavior, as can be seen in Table 3.4. In half of cases, if the probability does not allow evaluating the third rule, the output variation of the system would be zero. In the 40 percent of the cases (the 80 percent of the other half), only the third rule would be evaluated, so that the variation level would be exactly low positive. Finally, in the remaining 10 percent both rules would be evaluated so that the output would be the same as if probabilities were not used, that is, slightly below the low positive value.

In this way, we achieve this variability that we wanted so that not all agents will behave exactly the same. In addition, it will allow achieving some emotional stability in some cases, since some rules that cause a variation of the emotional attributes will stop being evaluated.

A similar effect is obtained during the aggregation process since degrees of matching determine the inclusion or not of the output fuzzy sets of every rule in

```
Center = (Personality[EmotionalAttribute] -
EmotionalState[EmotionalAttribute]) * ConvergenceSpeed
```

Figure 3.13: Updating the membership function of the `Personality` label

the aggregation process.

3.5.2 The dynamic evolution mechanism

The dynamic evolution mechanism performs a double function. First, it adjusts the probabilities associated with every rule of the fuzzy system. This variation of the probabilities modifies in time the set of rules more unlikely to be evaluated, so that the agent's behavior can be guided through a certain conduct. For example, if the level of *anger* of an agent has increased several times recently up to reaching a very high intensity level, the dynamic evolution mechanism will be able to decrease the probabilities associated with all the rules belonging to the fuzzy rule-based systems of all the other emotional attributes so that it will lead the agent to this kind of aggressive behavior. In this way, the rule probabilities will be automatically adjusted according to the flow of the conversation.

The second purpose of the dynamic evolution mechanism is to update the consequents of dynamic rules. If we remember, this kind of rules was identified by the consequent `Variation:Personality`, which allowed dynamically adjusting the level of variation of the emotional attributes according to the agent's personality and his current emotional state. The aim of these rules is to make emotions decay over time, bringing the agent near his equilibrium, defined from his personality. In this way, the dynamic component of the system updates, for every emotional attribute, the membership function of the `Personality` linguistic label defined on the basis of the values [`Center` - 0.075, `Center` - 0.025, `Center` + 0.025, `Center` + 0.075], where the `Center` value is calculated from the personality value and the emotional state for every emotional attribute using the formula shown in Figure 3.13, being `ConvergenceSpeed` the convergence speed to the equilibrium defined by the agent's personality.

For example, if we set `ConvergenceSpeed` equals 0.2, according to the formula of Figure 3.13, the `Center` will be included in the range [-0.2, 0.2]. In this way, the

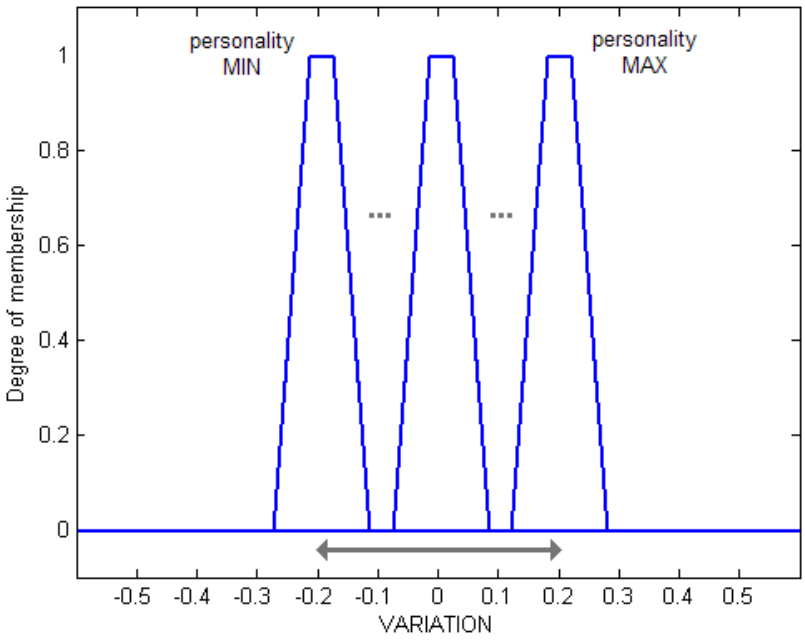


Figure 3.14: Variation range of the consequents of dynamic rules

membership function of the `Personality` linguistic label will be situated between the minimum and maximum membership functions defined, for this convergence speed, by the parameters [-0.275, -0.225, -0.175, -0.125] and [0.125, 0.175, 0.225, 0.275], respectively, as it is shown in Figure 3.14. The higher this parameter of the system is, the faster the agent's emotional state converges to the equilibrium defined by his personality. Moreover, the far away the agent's emotional state is from his equilibrium, the greater the level of variation is.

Therefore, according to the agent's current emotional state, the dynamic evolution mechanism of the system updates both the probabilities associated with rules and the consequents of dynamic rules in order to lead the agent to a particular behavior or to his equilibrium, depending on the specific situation.

3.6 Results

In this section we will show the results of some interaction tests that were carried out in our laboratory. We interviewed a virtual simulated patient using seven complete dialogs that had 9.4 natural language sentences on average. These dialogs were extracted from real doctor-patient conversations provided by the *IAVANTE Foundation*. Figure 3.15 shows how these sentences were typed on the user interface. We compared the emotional attribute variations produced by each sentence of the seven dialogs to the reactions that most of the people would have in those situations. The obtained results were the following. In 71.8% of the cases, the generated emotional behavior was the expected one, although the variation levels of the consequents of the rules had to be increased so that each emotional attribute could be able to achieve the maximum intensity during a complete dialog (some attributes had achieved 65% at the most for all the dialogs). The expected behavior was not 100% of the cases because we want the system to be non-deterministic, so that the result will not always be the same. For that reason, the remaining 28.2% corresponded with little variations opposed to the expected ones. On the other hand, despite the dynamic evolution mechanism, the initial probabilities had to be carefully chosen to avoid some undesirable situations, like fluctuations between inconsistent emotions. For example, the attribute *anger* initially experimented several fluctuations when the doctor asked about several symptoms that the patient was not suffering from.

In short, our conversational agent behaved reasonably well in terms of emotional behavior, naturally and with variability, in the conducted tests.

3.7 Conclusion and future work

In this chapter we have studied the emotional state of embodied conversational agents, one of the most essential components for the development of intelligent conversational systems that behave as humans do. We have reviewed some of the existing models for representing and controlling the emotional state. Moreover, we have analyzed the specific requirements of our problem (the development of a virtual simulated patient for training primary health care medical students)

Figure 3.15: User interface of the Virtual Simulated Patient

and we have seen how current models cannot cover all these requirements. For that reason, it has been necessary to develop a new control system that improves certain components included in other different models and incorporates some innovatory features as well. Finally, we have carried out several tests in order to measure the accuracy of our system. In general, the obtained behavior has been quite natural, although some little variations of the emotional attributes opposed to the expected ones have been detected. Besides, in spite of being applied to the particular domain of the virtual simulated patient, our system is easily adaptable to the rest of domains.

Regarding the future work, we have to study in depth how the different behavior patterns of human being defined in psychology evolve in the course of time. The objective is to carry out a natural adjustment of the probabilities of the rules because at this moment, as we mentioned before, they are kept constant along the conversation.

We also need to work on emotion filtering, since it will not always be suitable that agents show their emotions in the way they feel them. In some situations,

they will be interested in hiding them or pretending different emotions instead, according to their social context or personalities. For example, in our application it could be interesting that if a child patient went to the surgery with his mother, his behavior could be affected by the expressions that she showed. If he saw his mother become very sad or cry, he could worry if he did not really understand what is happening to her.

Finally, although the initial 2D model for the graphical representation of patients was not able to show several emotions at the same time (only the one with the higher intensity value was selected), the new 3D model do already allows the simultaneous appearance of emotions. However, we should analyze which emotions are not joined naturally in order to make a set of rules to avoid them.

This page intentionally left blank

4

Closed-domain virtual assistants

Since its beginning in 1969, the Internet has grown quickly, especially over the past few years. Companies and organizations store more and more information about themselves online. Sometimes, that information is not well organized. Other times, the huge volume of available data makes useful information difficult to be found. The result is that users have to waste their time looking for what they want to know using the traditional menu-driven navigation and keyword search that most websites provide. This is a critical issue because it decreases users' interest about companies. In order to avoid this problem, in this chapter we propose a methodology and a framework for designing closed-domain virtual assistants, which are, considering first results, an ideal alternative to help users find, not only the information that they are looking for, but also some related information which could be of the highest interest.

4.1 Introduction

With the arrival of the digital era and the development of the Internet over the last years, there are more and more companies that store all the information about them in a digital medium and publish it on the Internet, so that everybody around the world can have access to it. Traditionally, when people want to find some information about a company, they surf the Net and go to the website of that company, where they navigate through its menus and use keywords to look for what they want. However, due to the big amount of information, it gets very difficult for companies and organizations to structure their contents in a logical way so that they can be accessed using only one or two clicks or touches. Nowadays, this objective is usually a utopia because users have to waste their time looking for what they want to know using the traditional menu-driven navigation and keyword search that most websites provide. This decreases users' interest in surfing websites and hence in finding out about companies. If the information to handle is small, this problem can be resolved changing the structure of the contents and making the access easier. However, as the information grows, it is more and more difficult to provide users with an easy and fast access to all that data. Therefore, the necessity of new ways of accessing the information becomes evident.

First of all, we must stop to think why this problem is emerging, that is, why users have to spend a lot of time wandering through websites to get some useful information. There are many different reasons. The first one arises from the diverse nature of the Internet. Nowadays, there are millions and millions of websites about different topics. The main difficulty from a surfing point of view, although it is also one of the most valuable features of the Web, is that each website is different from the others. For this reason, there is a learning curve when you visit a website for the first time, and the steepness of the slope depends on how well organized the contents are and how skillful the user is. This is the second problem, there are many different kinds of user. Some of them are used to working with new technologies, and they can find the information that they are looking for in a website very quickly, even though they have never visited it before. However, other users find it very difficult to navigate through

a website using menus to discover new information, and they need somebody to support them. Another pitfall is the fact that companies store more and more information. Perhaps at the moment they do not have any problem, but in the future, as the company builds up, it is very likely that they do not know how to organize such an amount of information so that users have a fast and effective access to it. Eventually, there is another complicated problem that we should try to resolve. Sometimes people are not really sure about what they are looking for, so we cannot want them to choose a category from a menu because they do not know it.

All these problems that we have just mentioned make evident the necessity of an efficient and effective mechanism for organizing and accessing the information of a company when it becomes very big. That new system must have certain features. First, it must be real time. Users' time is highly valuable, so if they had to wait every time they wanted to get some information, the system would be useless. Another important feature is effectiveness, that is, the system must achieve the goals and objectives that users want. In this sense, the quality of the results must be high. Moreover, the communication between the system and the users must be natural, allowing them to ask natural language questions, that is, complete sentences rather than simple keywords. Finally, the interface of the system must keep as simple as possible so that everybody can use it, regardless of their computer skills. To sum up, the aim of the system must be to make the access to information easier for everybody.

The rest of this chapter is organized as follows. In Section 4.2 we begin with presenting some related work involving the use of intelligent systems for supporting users in surfing websites. We then present our Virtual Assistant in Section 4.3, describing its main features and making the advantages that it provides clear. We explain the architecture of our system in detail, as well as the different modules which it is made up of. Then, in Section 4.4 we describe the user interface and its functionality. Afterwards, in Section 4.5 we present the results of the first five years of Elvira [1], our Virtual Assistant for the University of Granada [2] (Spain). Finally, in Section 4.6 we conclude the chapter and provide some directions for

[1] http://tueris.ugr.es/elvira/?lang=en
[2] http://www.ugr.es/

future work.

4.2 Related work

An Embodied Conversational Agent (ECA) is an intelligent system represented by a character which is able to engage in conversation with a human being. This technology makes human-computer interaction much friendlier and can be used in any situation where there is a human-to-human interaction, with the ECA playing the role of any of them. Nowadays, ECAs are being more and more applied to many different fields. Entertainment, tourism, e-learning, e-commerce and medicine are just some examples of application domain. In this section, we focus on web navigation assistance and we make a chronological review of some intelligent systems that support users when they surf a website looking for some information.

As far back as the mid-nineties, people became seriously concerned about the recent explosive growth of the Web. Lieberman (1995) pointed out the necessity for some sort of intelligent assistance to a user browsing for interesting information. He introduced a behavior-based interface agent, Letizia, which operated in tandem with a conventional web browser. It tracked the user's browsing behavior (following links, initiating searches, requesting for help) and attempted to anticipate items of interest by doing concurrent, autonomous exploration of links from the user's current position. In this way, the user did not need to provide an explicit initial goal or evaluate the previous searches as successful or unsuccessful, but they were inferred from the user's actions using a simple set of heuristics. Thus, the system suggested a list of links to other documents, determining a preference ordering of interest among them. It also provided a reason for having chosen those recommendations, which decayed over time. However, Letizia did not have natural language understanding capabilities, so its content model of a document was simply a list of keywords.

Armstrong et al. (1995) also became conscious of the necessity of dealing with the growing flood of information available on the Web. For this reason, they designed another information seeking assistant called WebWatcher, which recommended hyperlinks given the current web page viewed by the user and her

information goal. They used a vector of boolean features to represent pages, links and goals. Each feature indicated the occurrence of a particular word within the text that originally defined these three attributes. Thus, the user introduced a set of keywords to indicate what information was going to be sought, and the system then returned a copy of the current web page, highlighting some especially promising hyperlinks. In addition, WebWatcher could follow users during their navigation, recommending new links as they accessed new web pages.

Yan et al. (1996) proposed a system for automatically classifying the visitors of a website according to their access patterns. This system periodically examined user access logs to discover clusters of users who accessed similar pages. Some of these clusters were not apparent from the physical linkage of the pages. This information could be used to improve the organization of the hypertext documents for navigational convenience, or to dynamically suggest links to navigate. In this way, when a user requested a new page, the module tried to classify her current partial session record against one or more of the categories obtained offline. The top matching categories were identified, and links to unexplored pages contained in these categories were inserted at the top of the page shipped back to the user.

Micarelli and Sciarrone (1996) presented Hypercase, a system for guided knowledge navigation in a hyperspace using Case-Based Reasoning (CBR) and neural networks. All possible thematic paths were defined by an expert on the domain and grouped according to the associated goals. At the beginning, these standard paths were hidden to the user, who was allowed to ask for help at any time. In that case, the system tried to find the closest standard path to the partial path followed by the user. Afterwards, a graph was used to give the user the right suggestions for the next step in the navigation.

Microsoft Agents (Ball et al., 1997) were designed to enhance the user interface of applications and web pages with interactive personalities in the form of animated characters. These characters could move freely within the computer display, show text, speak aloud, and listen for spoken voice commands. Although they were well known as helpers in Microsoft Office, they did not make a real success perhaps because they were too invasive sometimes and did not exhibit a very complex behavior. Nonetheless, they sufficed for helping inexperienced users in many situations.

Jaczynski and Trousse (1998) presented a browsing advisor for the Web called Broadway (BROwsing ADvisor reusing pathWAYs) which followed users during browsing sessions to infer their goals and then may propose some recommendations of potentially relevant documents to visit next. It used case-based reasoning to reuse past navigations with a time-extended situation assessment, which represented not only the current state of the observed navigation but also its past sequence of events. In this way, the recommendations were based mainly on the similarity of ordered sequences of past accessed documents, and they were presented through a list of URLs with a percentage rate representing an estimated relevance. Broadway operated in two modes, recommendations upon request, that is, only updating the recommendations when the user asked for help, or continuous recommendations, updating them automatically after each navigation move.

Wexelblat and Maes (1999) proposed a set of tools, called Footprints, to support undirected web browsing. Footprints was based on the concept of interaction history and the notion that the work done by past users can be important to helping current users solve problems such as navigation in a complex information space. A very good example is that buying a book is not the same as borrowing it. It is the same book (same words, pictures, and organization), but the borrowed book has additional information. A brand new book does not have any note in the margin, highlights, underlines, or dog-eared pages. Furthermore, the borrowed one reflects its history because it opens more easily to certain places once it has been used. These traces should be accessible to future users who could take advantage of the work done in the past to make their own problem-solving easier. With this aim, Footprints showed the traffic through a website using a graph in which nodes were documents and links were transitions between them. These were not all the documents and transitions, but only the ones that people had actually visited or used. These user-created transitions were as important as the static links provided by pages because they revealed user's models of how information should be connected. Footprints also showed paths, which were coherent sequences of nodes followed by a user, and annotated pages including the percentage of users who had followed each link. Moreover, both pages and paths could be commented by users.

Cockburn et al. (1999) made WebView, an add-on window to Netscape Navigator. It provided facilities for page identification and display organization. These included miniaturized thumbnail images of visited pages, shortcut link menus to navigate directly to subordinate pages, a dog-ears metaphor for bookmarking and page identification, and two organization schemes for displaying pages either by temporal properties or by structural relationships.

Cassell et al. (2000) created a very complete example of ECA called REA (Real Estate Agent) which played the role of a real estate salesperson who interacted with users to determine their needs, showed them around virtual properties, and attempted to sell them a house. REA had an articulated graphical body which was projected on a screen and could sense the user through cameras mounted on top of that screen. Furthermore, a computer run the graphics and the conversation engine, and several other computers managed the speech recognition and generation. Therefore, REA needed a lot of sensors and computational resources so it was not portable.

Abbattista et al. (2004) presented SAMIR (Scenographic Agents Mimic Intelligent Reasoning), a framework to build intelligent agents for the Web. They used an online bookstore as the application domain. SAMIR consisted of a 3D face which was animated to exploit expressions which were perceived by the user, a custom version of the ALICE (Artificial Linguistic Internet Computer Entity) chatterbot [1] to chat with the user, and a kind of classifier system to deal with the problem of keeping conversation and face expressions coherent with each other. The architecture had two main modules. The event interpreter converted all the events occurring in the course of the user interaction into parameters suitable to be processed by the behavior generator, which generated, according to a set of behavior rules, the character behavior as well as its emotional response, inducing correspondent facial expressions expressed in an abstract manner by MPEG-4 facial animation parameters (FAPs).

Kim et al. (2005) proposed an information retrieval assistant, called CHATTIE, which used natural language to support users in an e-commerce website. It had a client-server architecture and focused on fast response to users' queries and easy integration with outer information provision systems such as conventional

[1] http://www.alicebot.org/aiml/aaa/

information retrieval and relational database management systems. CHATTIE performed a linguistic analysis of the user's query with the help of three coordinated agents: a chat agent, for simple dialog queries (e.g. greeting or abuse), which used a database including a large collection of queries and associated actions; a question-answering agent, for informative simple context-independent queries (e.g. frequently asked questions), which used lexical-syntactic patterns; and a goal-oriented dialog agent, for context-dependent dialogs (e.g. ticket reservation dialogs), which used a frame-based method to fill slots. In this way, they generated an action which included an answer to the query and additional information to interact with the outer information system.

Kimura and Kitamura (2006) identified the following problem. AIML (Artificial Intelligence Markup Language) is an XML dialect for describing conversational scenarios for ECAs (Wallace, 2003). This language specifies pairs of patterns and templates, so the agent answers the template associated to the pattern that best matches the question. Traditionally, programmers use AIML to create conversational rules by hand, considering the contents of web pages. However, every time a page is updated, related conversational rules have to be modified, and this is a problem when many pages are frequently modified. In order to avoid this problem, they described a scenario to explain web pages, so that agents could understand them. They used RDF (Resource Description Framework) to represent the semantic contents of web pages. Besides, they used an RDF query language SPARQL (Simple Protocol And RDF Query Language) and an extended AIML language, which incorporated SPARQL queries, to make agents talk according to those contents. In this way, when a user asked a question, the server searched its AIML file for a conversational rule that matched that question. If a rule was found, the server submitted a SPARQL query in the rule and extracted some information from the designated RDF file. It then inserted the extracted data into the template in the rule and returned the answer and a URL with additional information to the client.

Khentout et al. (2007) made a comparative study of some of the navigation assistance tools that can be found in this chapter. They stated that these tools were difficult to compare because of the variety of goals and contexts. However, they took into account the following features: the visualization technique used by

the systems (trees, maps...), that is, if they allowed different detail level or zoom, if links or contents were annotated, the capacity of the systems to react to different user's interactions (degree of help), if they allowed multi-sites or only a specific hypermedia, if they could change and evolve according to different strategies, and finally the independence from browsers. They also implemented NaVir, a computer-aided system for web navigation which facilitated the learning process within a platform for distance education on the Web. It collected the different URLs visited by users, and it built and interacted with the graphical map and the management of the navigation time. Besides, they used a proxy server to retrieve the addresses of those pages in order to guarantee the independence from the browser.

Pilato et al. (2008) proposed an intelligent tutoring system to support students in the learning of the Java programming language. When a student asked a question, the system looked for the concept that best matched that question and then created a backward path from that concept to the student's current knowledge state. The concepts and their relations were stored in a hierarchical ontology. Relations could be strong (prerequisites) or weak (something related). Most of the subjects were mandatory, although some of them were optional, being useful for some students to explain some concept. Each document had a list with the most frequent non-stop words, which could be used to cluster documents and also evaluate if a student knew the concept referred by a document. In order to identify the concept to which the student's first question referred, and evaluate students and see what they already knew, they used LSA (Latent Semantic Analysis) techniques. The learning path was the list of all the unknown nodes. For the user interface, they clustered documents, grouping the ones that referred to a same subject and putting closer which were more similar.

In addition, some commercial assistants can be found on the Web such as the ones of Artificial Solutions [1].

As we can see, many of the existing systems, especially those from earlier years, lack a virtual character which can engage in conversation with users, making the interaction process friendlier. What is more, some systems do not allow users to ask natural language questions. In addition, they should take account of

[1] http://www.artificial-solutions.com/

the context during the dialog, so that users could omit the implicit words in the conversation. Moreover, some others systems focus on the navigation history, that is, they only try to organize the pages that have been already visited, instead of recommending related web pages to continue the navigation. These and other problems make the navigation process much more difficult. For this reason, in this chapter we propose a methodology and a framework for designing closed-domain virtual assistants which can be integrated into every existing website, and which cover all the features mentioned before.

4.3 System design

In this chapter we present a framework for designing closed-domain virtual assistants, that is, embodied conversational agents that help people find useful information about a topic in a website. These virtual assistants are real time intelligent systems because users do not want to wait for getting the information they want. They engage users in conversation using natural language, as if they were real assistants. This way of communication is much better than the one offered by traditional websites, where users can only click on static menus and do searches specifying some keywords, and it is perfect for users without computer skills. In addition, users can omit some words if they are implicit in the conversation, so these virtual assistants must be context-aware. Finally, another important feature is that they do not only answer people's questions but also offer related information or recommendations that lead users and keep the conversation going.

Our system has been designed using a client-server architecture with four main modules, as can be seen in Figure 4.1. The Natural Language Understander recognizes the question asked by the user. The Dialog Manager generates an abstract answer to that question. The Emotional State Controller manages the emotions of the virtual assistant. Finally, the Communication Generator retrieves and adapts the abstract answer generated by the Dialog Manager to make a specific final answer. Next, within this section, we will explain the architecture of the system and each module in detail. Moreover, we will show all the features of the user interface, which allows users to interact with the system.

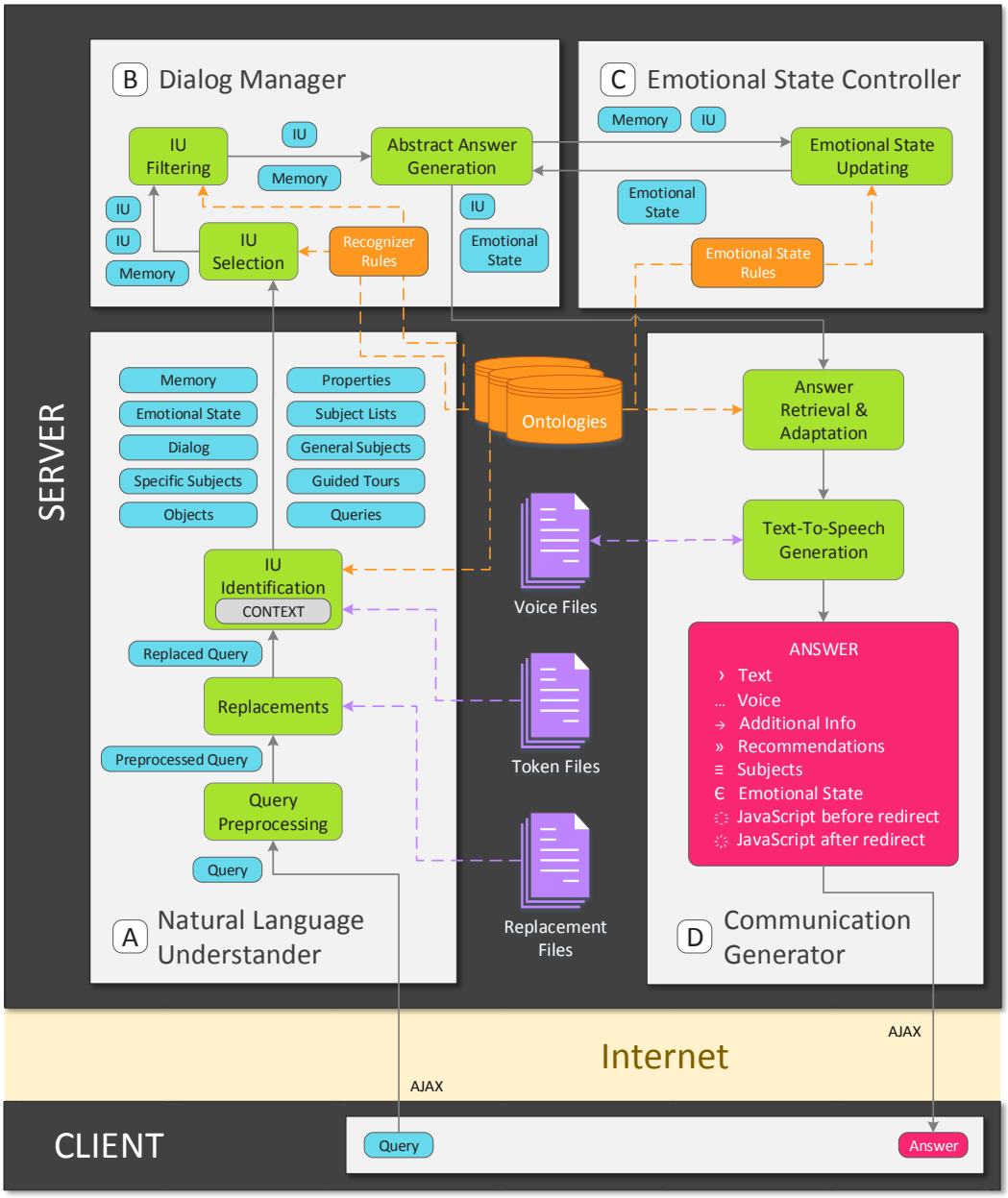


Figure 4.1: Architecture overview

Making the access to information easier for any kind of user must be one of the main objectives of our system. Therefore, we must reduce the prerequisites for the host machine to the minimum. For this reason, the proposed system has

a client-server architecture. In this way, all the computation is done on the server side and users do not need to install any program in their computers. They only need a web browser with Flash [1] capabilities to see the face of the virtual assistant, and JavaScript enabled in order to send AJAX (Asynchronous JavaScript And XML) requests to the server and display the answers generated by the system.

Figure 4.1 also gives us an overview of how the system generates an answer for a specific question. First of all, the user's question is taken from the client to the server using the AJAX technology. Once the server has received this query, it is processed by the Natural Language Understander (NLU) in order to identify the information units (IUs) to which it refers (in Section 4.3.1 we will explain this concept of information unit in detail). Then, the Dialog Manager (DM) receives a list, grouped by the different information categories, with the IUs that have been identified by the NLU. Using a set of rules, the DM generates another list with those IUs that really answer the user's question. Afterwards, the DM filters this new list, removing all the ancestors to keep the most specific IUs, and then it creates an abstract answer. This contains the new emotional state of the virtual assistant, which is updated by the Emotional State Controller (ESC). Next, the Communication Generator (CG) receives the abstract answer and retrieves and adapts a specific answer, which is made up of several elements. First, it contains not only the text of the answer but also the URL of a sound file, since the virtual assistant provides a spoken answer. Moreover, it includes the URL of a web page which contains some information which is complementary to the answer, and a list of recommendations or subjects about related topics that can be used to continue the conversation. It also contains the new emotional state and it may include some JavaScript code to interact with the user before or after redirecting her to a new page. Finally, the generated answer is sent back to the user at the client side using AJAX.

Regarding the technologies that we have used for the implementation of the system, we can mention the following ones. On the server side, we use the Java programming language for the core of the framework, the Protégé [2] (Noy et al., 2001) program to store and edit the knowledge by means of ontologies, the

[1] http://www.adobe.com/products/flash/
[2] http://protege.stanford.edu/

Apache Jena [1] library to handle this knowledge, using a rule language to make queries and retrieve information, and the Nuance Loquendo TTS (text-to-speech) [2] to transform the answers provided by the virtual assistant into spoken dialog. On the client side, as we have already mentioned, the communication with the server is carried out using the AJAX technology, and the user interface uses Flash for displaying the avatar that represents the virtual assistant and the jQuery [3] JavaScript library for handling and animating the different components of the interface.

4.3.1 Knowledge representation

The system uses two types of knowledge, each one with a different functionality. On the one hand, it uses several plain text files containing thousands of regular expressions to recognize the user's question. On the other hand, it uses ontologies to store all the information related to the different concepts, answers, recognizer rules and emotional state rules. These two kinds of information are detailed in this section.

As we said before, all the information related to the concepts and the answers provided by the virtual assistant is stored by means of ontologies. According to the definition of Gruber (1993), an ontology defines a set of representational primitives with which to model a domain of knowledge. These primitives are typically classes (or sets), attributes (or properties), and relations (or connections among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. In other words, an ontology is essentially a formal representation of a set of concepts within a particular domain and some relations established between those concepts.

Our ontologies are based on entities that we have called information units (IUs). An IU is a piece of information about a specific concept which has a meaning by itself. It not only includes the definition of the concept but also some

[1] https://jena.apache.org/

[2] http://www.nuance.com/for-business/customer-service-solutions/loquendo-small-business-bundle/index.htm

[3] http://jquery.com/

meta-information such as its name, a URL where more information about it can be found, and so on. We have defined six different types of IUs. In Section 4.3.3 we will explain the reasons for treating them in a different way. For the time being, we will give a brief description about each kind of IU. Note that, as we will show in Section 4.5, the methodology and the framework presented in this chapter have been used to design Elvira, a virtual assistant to provide information about the University of Granada, in Spain. For this reason, all the examples that we will show below belong to a university domain.

Objects Any individual of the ontology (or the domain, in general) having a fixed structure shared with other individuals of the same class is considered an object. In this way, we can define templates for the objects of our domain, specifying the names of their properties, so that we do not have to define the same properties every time we create a new object. The degrees, the people with high-ranking position at the university, the faculties and schools, the vice-chancellors' offices, the secretariats, or the services are just a few examples of the hundreds of objects present in the ontology. Figure 4.2 shows all the information related to the object "*Degree_Medicine*".

Properties Each one of the features of an object is a property. They are considered different from objects for two reasons. First, objects may have properties, but properties must not have either objects or properties. Second, this philosophy allows us to identify object names and property names independently, so if a question is ambiguous and only the name of the property can be identified, the system can urge the user to concrete the object. At the moment of writing this chapter, the ontology contained more than one thousand property individuals. If we consider for example the object "*International Project Office*", we can identify several properties such as the address of the office, the telephone number, the email address, the fax number, the website, the person in charge, and so forth.

Specific subjects The concept of subject is quite wide. They represent services offered by departments (in the broadest sense of the word). There are hundreds of specific subjects to provide information about the access to the university

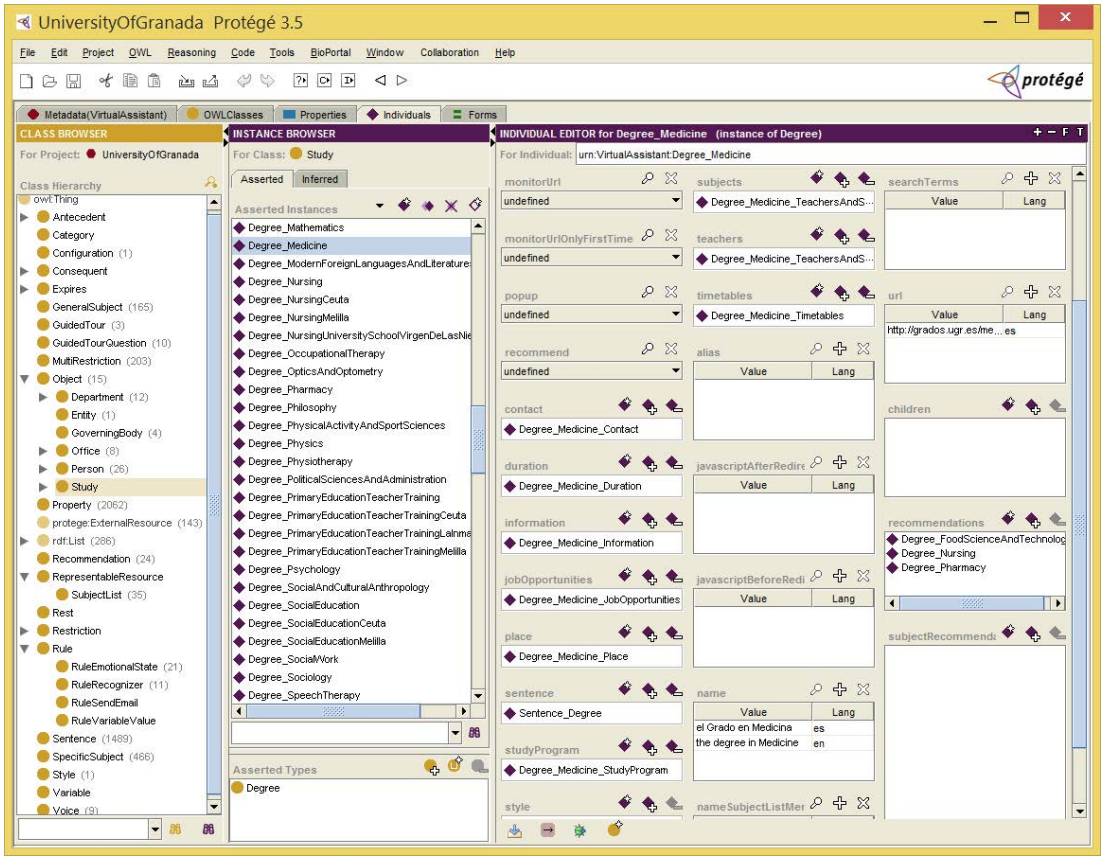


Figure 4.2: Editing the ontology of the University of Granada with Protégé

education, the different scholarships for students, the sport courses offered by the sports activity center, forms, regulations, projects, and so on and so forth.

General subjects Whereas specific subjects offer information about a well-defined particular subject (the "*Erasmus Scholarships*", for example), general subjects are used when it is not clear what the user is asking ("*Scholarships*", in general) since there are several possible contexts that can be applied. In that case, the user could be interested in one option or another ("*Student Scholarships*", or "*PhD Student Scholarships*"), which are intended for mutually exclusive user groups. Therefore, the aim of general subjects is to find out the appropriate context for each user. In Figure 4.3 we can see the relations that some of the subjects have in the ontology. Note that some links and all the information about

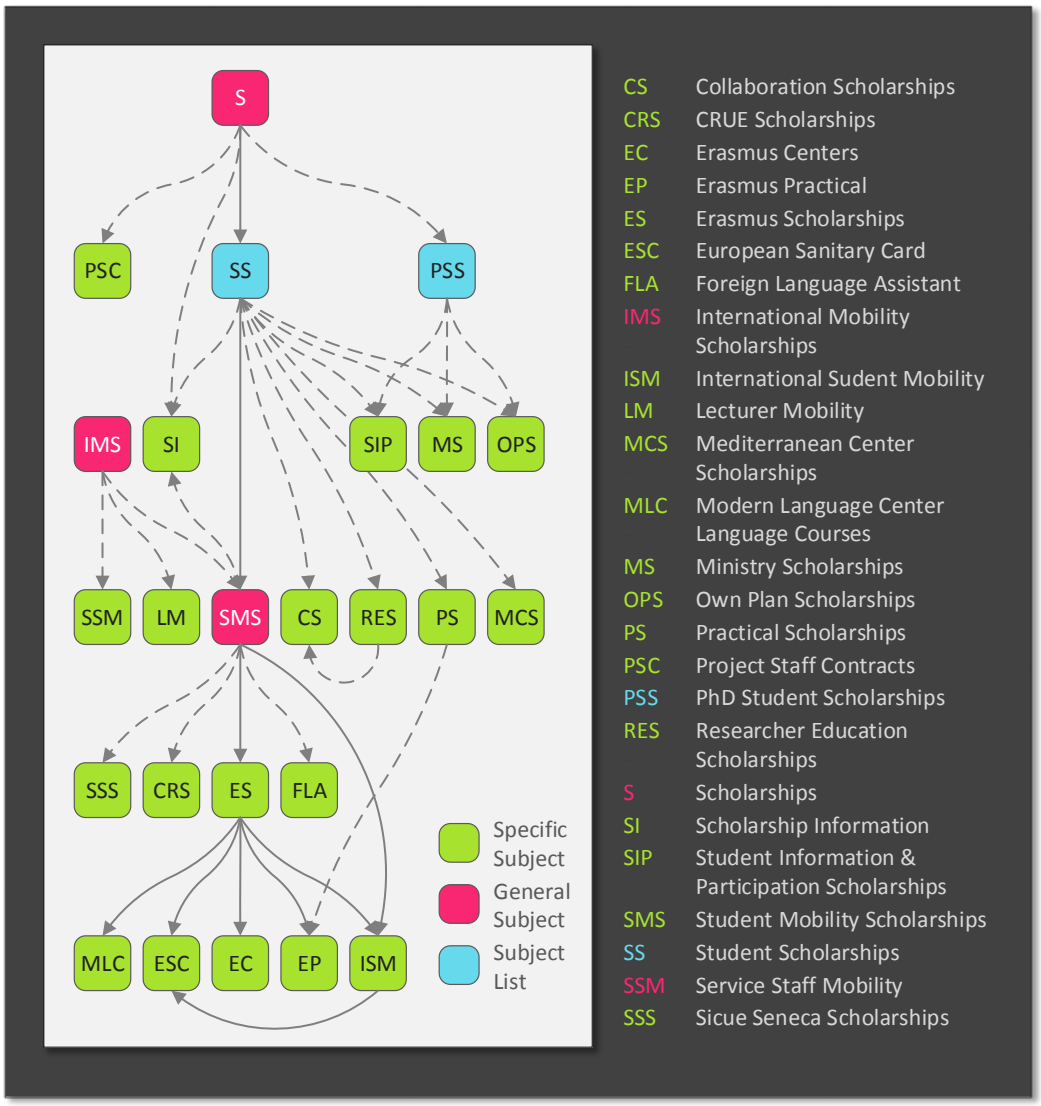
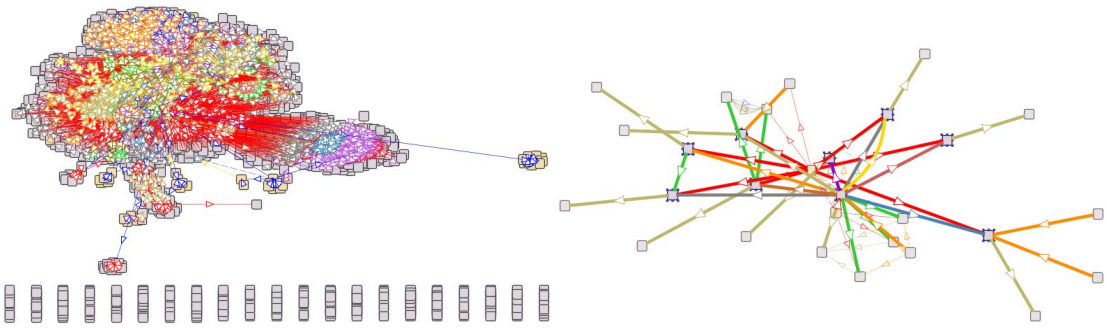


Figure 4.3: A subset of all the subjects included in the ontology

the objects and properties related to those subjects have been omitted for the sake of simplicity.

Subject lists When it is perfectly clear what the user is asking, and there are several options that could be interesting, subject lists are used. For example, a



(a) Overview of all information units (b) Neighborhood of "*Degree_Medicine*"

Figure 4.4: Connections in the ontology of the University of Granada

user asking about the "*Student Scholarships*" could be interested in more than one option because the different kinds of student scholarships ("*Collaboration Scholarships*", "*Ministry Scholarships*", "*Student Mobility Scholarships*", and so on) are not intended for mutually exclusive user groups.

Guided tours A guided tour leads users step by step throughout a specific path determined in advance. Using rules, it is possible to design processes specifying the different stages of which they are made up. For example, there is a tutorial that shows new users all the capabilities of the system, another allows sending suggestions, and so on.

All these IUs are not isolated in the ontology, but they can be connected with each other, as can be seen in Figure 4.4. Consequently, we can see our ontology as a kind of graph in which the IUs are the nodes and the connections between them are the edges of the graph.

The other type of knowledge that is handled by the system is all the information needed for the recognition of the different questions. This data is stored using plain text files which contain tokens associated to the IUs, representing different patterns or regular expressions for all the possible questions about them. Figure 4.5 shows some examples for the general subject "*Bologna Plan*".

In order to simplify the matching process between the user's question and the regular expressions included in the token files, we use another file, called the replacement file. In Figure 4.6 we can see an excerpt of that file. As token

$BolognaPlan\_GS : (bologna|bologna\_plan)$

$BolognaPlan\_GS : (new\ (((universit(y|ary)|educational)\ )?$
$(degree(s)?|plan(s)?|study\_plan|qualification(s)?))$

$BolognaPlan\_GS : (european((\ higher)?\ education)?\ area|$
$european\ (space|area)(((\ for)?\ higher)?\ education)?|$
$EHEA|europe(an)?\ education)$

Figure 4.5: An excerpt from a token file

files, it includes a list with regular expressions about possible misspellings that can appear in the user's question, synonyms, or complex IU names, for example. Each one of these regular expressions has a word associated with it, which is used to replace the corresponding piece of text when it appears in the question. In this way, during the first step of the recognition process, the number of words of the question is reduced. This makes the processing of the token files faster since less operations must be done. It also increases the reliability of the understanding process because it reduces the ambiguity of questions and it simplifies the maintenance of the system.

Next, we will analyze each module of the architecture in detail, explaining how they use the knowledge stored in the replacement file, the token files and the ontology in order to carry out their tasks.

4.3.2 The Natural Language Understander

When a user asks a question, the first step that the system must take is to recognize what it refers to. In other words, it must identify the information units (IUs) that are referred by the question, in order to be able to generate an appropriate answer later on. This is the role of the Natural Language Understander (NLU). Its input is a natural language question, and its output is a list with all the IUs that are referred by that question. This list is later used by the Dialog Manager (DM) to determine what the virtual assistant has to talk about.

The understanding process consists of three steps, as can be seen in Figure 4.7. First, the user's query is preprocessed, that is, it is converted to lowercase and

$vice\_chancellor : vice(-|\ )?chancel(l)?or$

$aid : incentive(s)?|aids$

$email : e(lectronic)?(\ |-)?mail(s)?$

$ask : ask(s|ed|ing)?|request(s|ed|ing)?$

$search : search(es|ed|ing)?|find(s|ing)?|found|look(s|ed|ing)?\ for$

$what : waht|hwat|whta|wha|wht$

$when : (at)?(what|which)\ time$

$where : (what\ ((is|are)\ the)?)?(address(es)?|place(s)?|location(s)?|$
$localization(s)?)(\ of)?$

$how : what\ should\ (i|we)\ do\ for|what\ do\ (i|we)\ need\ for|how\ do\ (i|we)$

$ugr\_organization : univ(ersity)?(\ of)?\ granada|granada('s)?\ university|ugr|$
$ugranada|u\ g\ r|(this|your|the)\ (university|entity|$
$institution|organization)$

$information\_technology : (((university\ )?degree(s)?|plan(s)?|study\_plan|$
$qualification(s)?)(\ (of|in|for)\ )?)?$
$(((information\ )?technology)|$
$(computer(s)?\ science(s)?))$
$((\ (in|of))?\ ugr\_organization)?$

$june\_examination : (ordinary\ )?exam(ination|s)?(of)?\ (june|july)|$
$(june|july)\ exam(ination|s)?$

Figure 4.6: An excerpt from a replacement file

special characters such as accents, punctuation marks, or extra spaces are removed. During the second step, the Natural Language Understander uses the replacement files to reduce the language of the preprocessed query to a specific vocabulary so that it can be managed by the system more easily. In that way, it corrects some possible misspellings, replaces some words with synonyms and

some complex names with abbreviations, etcetera. Remember that the virtual assistant must be real time because users cannot wait for getting an answer to their questions. In the third step, the NLU uses the token files to generate a list with all the information units referred by the replaced query (objects, property names, specific subjects, general subjects, subject lists, and guided tours). This list also includes contextual information like the IU used to answer the previous question asked by the user.

Now we are going to see an example of query and how it is processed by the system. Let us suppose that a user asked the question "*What is the address of Information Technology?*". First, the NLU would preprocess the query, converting the question to "*what is the address of information technology*". After that, it would replace some expressions using the replacement file shown in Figure 4.6, giving as a result the query "*where information_technology*". This replaced query would be used by the NLU to look for the IUs in the token files. In this case, two objects ("*Information_Technology*" and "*Higher_Technical_School_of_IT_and_TE*"), and three property names ("*Entity_Address*", "*Department_Address*" and "*Center_Address*") would be identified and passed to the Dialog Manager, which would generate an appropriate answer. In this case, the answer would be "*Center_Address_Higher_Technical_School_of_IT_and_TE*".

In conclusion, the Natural Language Understander receives a natural language query from the user and generates a list with all the information units that match that query.

4.3.3 The Dialog Manager

The part of the system in charge of making decisions about what the virtual assistant must answer is the Dialog Manager (DM). The inputs of this module are the list of information units that match the user's question (previously generated and grouped by categories by the Natural Language Understander), the current emotional state of the virtual assistant (managed by the Emotional State Controller), and the memory (a record with all the questions asked by the user including the answers provided by the system). The output is an abstract answer that will be later retrieved and adapted by the Communication Generator (CG)

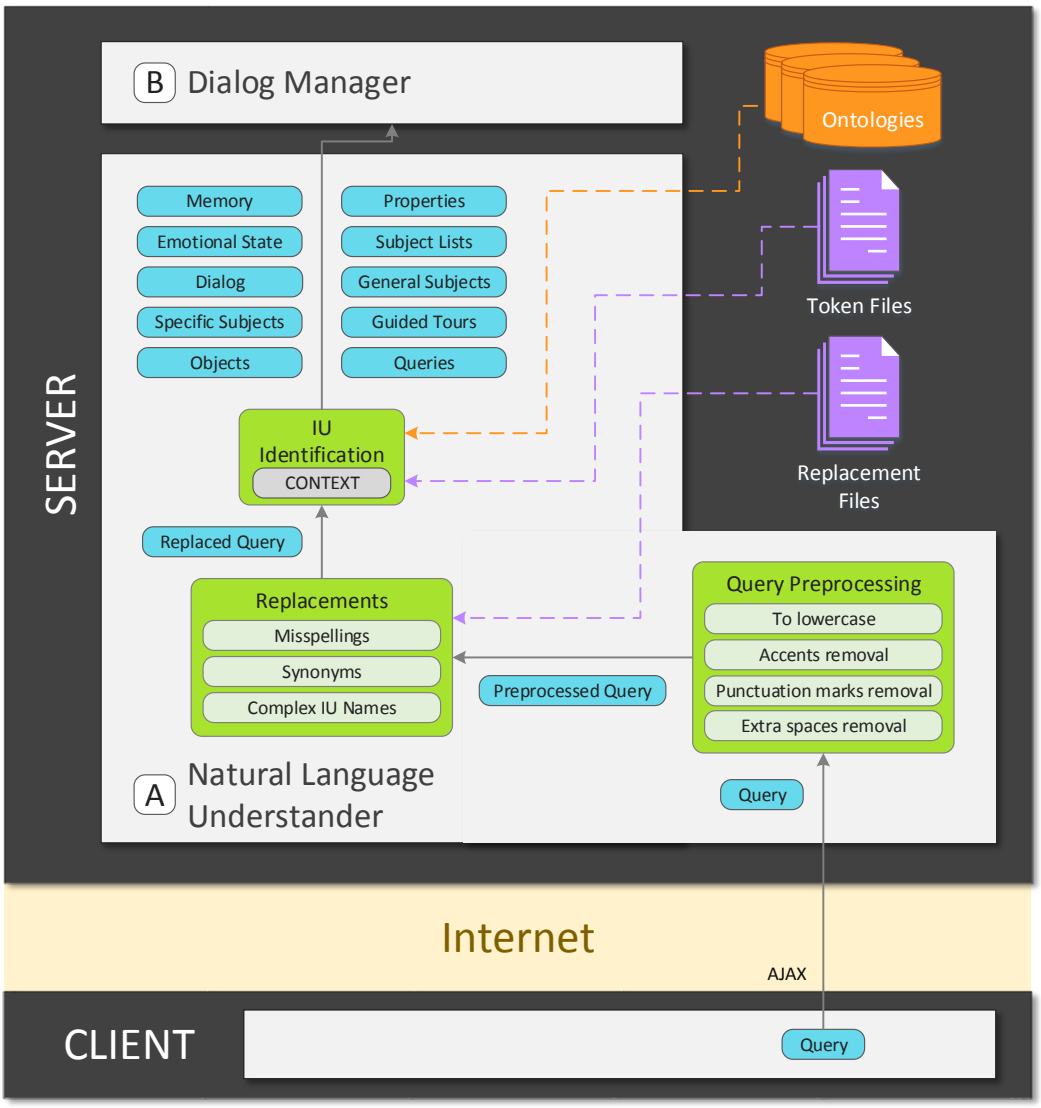


Figure 4.7: The natural language understanding process

to make a specific final answer.

With the aim of selecting the IUs that must be used to answer the user's question and including them into an output set, it is very important that the list of IUs identified by the NLU be divided into different categories. In this way, the DM can analyze them and carry out a rule-based selection process.

Each rule is made up of a set of antecedents and a set of consequents. When all the antecedents of a rule are true, that rule fires and all its consequents are applied. We have defined twelve different classes of antecedents and six types of consequents (although more could be defined if necessary).

Classes of Antecedents:

- `AntecedentAnswer`. If the current answer, that is, the list of IUs identified by the Natural Language Understander, contains a certain IU, or not.
- `AntecedentAnswerClass`. If the current answer contains an IU belonging to a certain class, or not.
- `AntecedentAnswerUrlPattern`. If the URL of the web page in which the user is matches a specific regular expression, or not.
- `AntecedentCulturalLevel`. If the virtual assistant has a certain cultural level, or not.
- `AntecedentDialogState`. If the conversation is in a specific state, or not.
- `AntecedentEmotionalState`. If the virtual assistant is in a certain emotional state, or not.
- `AntecedentEnvironment`. If there exists a particular aspect in the environment of the virtual assistant regarding its interaction with other individuals (agents or people) of the world, or not.
- `AntecedentHealth`. If the health level of the virtual assistant is within a certain interval, or not.
- `AntecedentMemory`. If the memory, which stores all the conversation record, contains a certain IU, or not. In other words, if the virtual assistant has already talked about a certain IU.
- `AntecedentPersonality`. If the virtual assistant has a certain personality, or not.

- `AntecedentPreviousAnswer`. If the previous answer contained a certain IU, or not.
- `AntecedentPreviousAnswerUrlPattern`. If the URL of the web page in which the user was in the previous question matches a specific regular expression, or not.

Classes of Consequents:

- `ConsequentAnswerClass`. It establishes certain priorities in the set of recognized IUs that were going to be used to answer the question. This allows removing all the IUs that belong to any class that is not listed in this consequent.
- `ConsequentEmotionalState`. It updates the values of the emotional attributes of the virtual assistant.
- `ConsequentOnlyAnswer`. It leaves a certain IU as the only one to be used to answer the question.
- `ConsequentRemoveAnswer`. It removes a certain IU from the set of recognized IUs.
- `ConsequentSendEmail`. It sends an email to a specific address.
- `ConsequentVariableValue`. It defines a certain variable and sets it to a specific value. This allows capturing information from the user and storing it in a record, or even changing the particular sentence of an IU that is used to answer the question.

According to the type of consequent they have, rules are classified in one of the following four types:

- `RuleEmotionalState`. Rules with a consequent belonging to the `ConsequentEmotionalState` class.
- `RuleRecognizer`. Rules with a consequent of the `ConsequentAnswerClass`, `ConsequentOnlyAnswer` or `ConsequentRemoveAnswer` classes.

- `RuleSendEmail`. Rules with a consequent of the `ConsequentSendEmail` class.

- `RuleVariableValue`. Rules with a consequent belonging to the `ConsequentVariableValue` class.

This classification allows evaluating each type of rule independently. For example, in Section 4.3.4 we will see how the Emotional State Controller evaluates all the rules belonging to the `RuleEmotionalState` class. All the other rules will be evaluated by the Dialog Manager and they will allow resolving conflicts that might appear when more than one IU has been identified by the NLU. In addition, they could completely change the answer provided by the virtual assistant. For example, a rule could state that if the query contains any unpleasant dialog, the virtual assistant must plead with the user to speak in an appropriate way. So, if a user asked "*show me the menu of the dining hall, idiot*", the NLU would identify the unpleasant dialog, and the DM would refuse to provide the requested information and it would restrict the virtual assistant to demanding respect. On the other hand, another rule could state that, if the query is polite and the NLU identifies an object and a specific subject, only the latter is selected by the DM. This would be the case of the query "*cut-off grades to get into medicine*", where the objects "*Medicine*" and "*Faculty_Of_Medicine*" and the specific subject "*CutOff_Grades*" would be identified by the NLU. Finally, if the output of the NLU did not contain any IU, the DM could search the query using the search engine of the organization.

Once the Dialog Manager has determined the output set with the information units that are going to be used to answer the question, it applies several filters using all the contextual information that can be extracted from the structure of the ontologies. These filters remove all the ancestors that appear at the same time as their descendants. The objective is to avoid a certain type of ambiguity that might be present in the question asked by the user. For example, in the query "*what mobility scholarships are there?*", the NLU would identify two general subjects, "*Scholarships*" and "*Mobility Scholarships*". Since the latter is more specific, the DM would select "*Mobility Scholarships*" to answer the question.

The remaining IUs (at least one) are used to generate the abstract answer that will be used by the Communication Generator to create a specific answer. An abstract answer is made up of a set of IUs, which are the concepts that we want to talk about. For example, we might want to answer using a particular subject, ask the user for clarification if more than one object or subject has been selected at the same time, and so on. This abstract answer also contains the new emotional state of the virtual assistant. As we will see in Section 4.3.4, this is updated by the Emotional State Controller from the filtered IUs and the information stored in the memory of the virtual assistant.

To sum up, firstly, the Dialog Manager analyzes the set of IUs that have been identified by the Natural Language Understander and selects, according to a set of rules, those IUs that are going to be used to answer the user's question. Next, those IUs are filtered in order to keep just the most specific ones. These, together with the new emotional state of the virtual assistant, are used by the DM to make an abstract answer which is sent to the Communication Generator to create a specific answer.

Next, before going into the details of the Communication Generator module, we will see how the Emotional State Controller manages the emotions of the virtual assistant.

4.3.4 The Emotional State Controller

The emotional state of the virtual assistant is managed by the Emotional State Controller (ESC) using the dynamic probabilistic fuzzy rules based system (Eisman et al., 2009a) described in Chapter 3. The inputs are the list of information units filtered by the Dialog Manager that are going to be used to answer the question asked by the user, and the information contained in the memory of the virtual assistant. The output is the new emotional state that will be included in the abstract answer generated by the DM.

As we saw in Section 3.3, the emotional state control system is based on two essential concepts: emotional state and personality. Emotional state is built on the basis of eight basic emotions that correspond with the following emotional attributes: joy, disdain, anger, fear, worry, surprise, sadness, and embarrassment.

```
AntecedentAnswer[Answer == ComplimentGoodLooking]
∧ AntecedentMemory[ComplimentGoodLooking == Zero]
⇒ ConsequentEmotionalState[EmbarrassmentVariation = Low_Positive]
```

Figure 4.8: Example of fuzzy rule to control the emotional state variation

The value of each attribute is a real number between 0 (total absence) and 1 (total presence). On the other hand, personality is a set of static features that allow carrying out a subjective assessment of the context of the world and the questions asked by the user. In addition, it can make the emotional state tend to an equilibrium, which will vary according to the type of personality. In this way, the virtual assistant will not keep a specific emotional state forever if no rules are fired but it will vary with every single question. Personality is defined, as well, on the basis of the eight emotional attributes described before. Six different personalities have been defined: anguished, depressive, hypochondriac, maniac, phobic, and normal.

Every emotional attribute is updated independently and in parallel by means of a set of fuzzy rules based systems. These have a set of input variables such as the personality of the virtual assistant, the type of question asked by the user, the memory, the flow of the conversation on the basis of a set of states, and so on. The output variable is the specific variation level to produce in each emotional attribute, from high negative to high positive.

In this way, by means of a set of fuzzy rules stored in the ontology belonging to the `RuleEmotionalState` class, the ESC updates each emotional attribute. For example, the rule shown in Figure 4.8 states that IF the selected IU to answer the question is `ComplimentGoodLooking`, AND this IU is not already present in the memory of the virtual assistant (it is the first time the user makes a compliment), THEN a low positive variation of the embarrassment emotional attribute is produced.

In addition, every rule has a probability value associated with it that corresponds to the probability of taking it into account at the current state of the system to calculate the emotional state transition. This allows the system to behave in a non-deterministic way as far as the emotional state of the virtual assistant is concerned.

4.3.5 The Communication Generator

The aim of the Communication Generator (CG) is to instantiate the abstract answer provided by the Dialog Manager (DM) in a concrete natural language answer easily understood by the user. In other words, once the DM has established what the virtual assistant is going to talk about, the CG looks for the specific words that are going to be used to answer the question. Whereas the abstract answer of the DM is language independent, the specific answer of the CG is not, and it may be expressed in different languages. In this way, the framework we propose in this chapter allows the design of multilingual virtual assistants.

The answers of the system are made up of different information categories, which are stored in the ontology next to each IU, and which we could see in Figure 4.1. The core of the answer is a piece of text (or an HTML fragment) associated to the IU selected by the DM. The information contained in this text could be always the same or not. Sometimes, with some IUs, the system always provides the same information, although each time the specific words (the concrete sentence) of the answer might vary. In other cases, the information is not always the same, and it may change depending on a set of constraints like the ones described next:

- `RestrictionHour`. It specifies a start and an end time (e.g. everyday, from 8:00 a.m. to 3:00 p.m.).
- `RestrictionDayOfWeek`. It specifies a start and an end day (e.g. from Monday to Friday).
- `RestrictionDayOfWeekHour`. It specifies a start and an end day, and a start and an end time (e.g. from Monday to Friday, from 8:00 a.m. to 3:00 p.m.).
- `RestrictionDayOfMonth`. It specifies a start and an end day of month (e.g. every month, from 1st to 15th).
- `RestrictionMonth`. It specifies a start and an end month (e.g. every year, from January to June).

- `RestrictionDayOfMonthMonth`. It specifies a start and an end day of month, and a start and an end month (e.g. every year, from January to June, from 1st to 15th).
- `RestrictionDate`. It specifies a start and an end day of a year (e.g. from January 1st, 2015 to June 30th, 2015).
- `RestrictionDateTime`. It specifies a start and an end time (e.g. from 8:00 a.m. on January 1st, 2015 to 3:00 p.m. on January 10th, 2015).
- `RestrictionUrlPattern`. It specifies a regular expression for the URL visited by the user to match.
- `RestrictionVariableValue`. It specifies a variable and a value (e.g.: $greetings == false$).

For example, in summer, the swimming pool of the university is open-air, but in winter, it is heated. Thus, if a user requests information about the swimming pool, the virtual assistant could provide different information according to the season in which the user poses the query, which is the information that she wants to know.

Restrictions are also useful to inform users about deadlines or events very quickly. For example, official registration periods, cultural events, latest news, and so on. This information can be included in the "*Welcome*" message of the virtual assistant and it can be configured to appear randomly during a certain period of time.

The textual answer about a particular IU is also transformed into speech using the Nuance Loquendo TTS. The whole process is shown in Figure 4.9. When a sentence needs to be converted to MP3, the virtual assistant generates an identifier for it using a hashing algorithm (e.g. MD5, SHA-1...). Next, it looks for this hash in a cache of previously generated MP3 files. This cache allows reducing the execution time of a query to the virtual assistant. If the sound file already exists, no additional steps are required, it will be retrieved from its URL. However, if the MP3 file does not exist, the system must generate it. For this purpose, a request is sent to a TTS web service, which will generate the new MP3

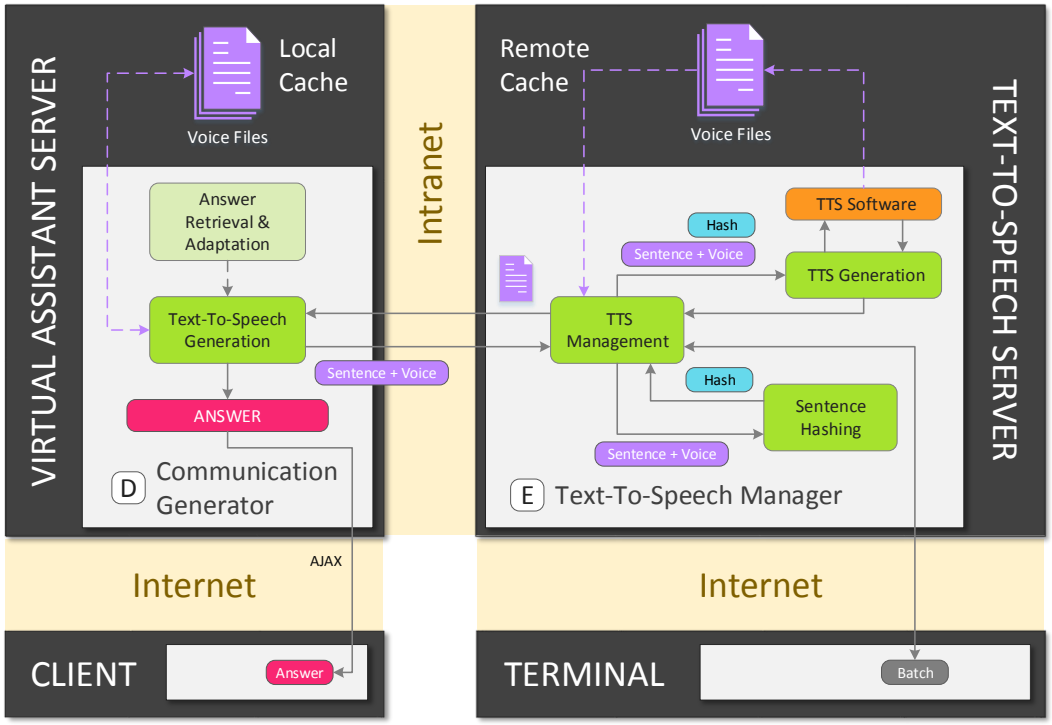


Figure 4.9: The text-to-speech generation process

using the desired language, voice and sentence parameters. This new sound file will be received by the virtual assistant, which will copy it to the local cache, ready to be served.

As well as the textual answer and the sound file about a particular IU, the virtual assistant also shows a web page where the user can find more information about that IU. In addition, it could provide two lists with some other IUs related to the answer, namely, the recommendation list and the subject list. In Section 4.4 we will show the difference between both lists in practice. For the time being, we can say that they are really useful because they might contain very interesting information to the user, who could be not conscious of its existence in a first moment. For example, a student might be interested in finding information about the different master's degrees that are offered by the University of Granada. With this purpose in mind, she could ask the question "*I'd like to see the list of the official master's degrees*", and the virtual assistant would show her that list.

However, maybe the student does not know that some kinds of scholarships could pay her registration fee of the master degree. In this case, a recommendation would be ideal.

In short, the Communication Generator instantiates the abstract answer provided by the Dialog Manager in a concrete natural language answer that can be easily comprehended by the user.

4.4 The User Interface

There is a famous quotation from Albert Einstein which says "*make things as simple as possible, but no simpler*". This should always be the maxim when designing the user interface of an application. Following this line of work, we have designed several interfaces in the course of time, as it is shown in Figure 4.10. Initially, the virtual assistant was represented using the 2D model that appears at the top of Figure 4.10a. This model could move the head, blink, and also speak. This made the conversation much more natural and similar to a real dialog. However, it was not able to feel emotions such as joy, anger, surprise, or sadness, and the move of the avatar was not completely natural. In addition, although this top-sliding interface could be minimized (just leaving a tab at the top-right corner of the website), it was a little bit invasive sometimes. For these reasons, we designed the new user interface with a 3D model that can be seen at the bottom-right corner of Figure 4.10b. This new virtual assistant did allow showing emotions, answering in different languages, and adapting the answers provided to make them consistent with the emotions that it felt. Moreover, this new interface was less invasive and gave more importance to information.

Figure 4.10b also shows an example of answer for the "*Welcome*" information unit. As we can see, it includes a dynamic recommendation list which allows the virtual assistant to offer new information related to the answer. In this example, the conversation with the user has just started, so Elvira introduces herself and urges the user to ask any question about the University of Granada (UGR) or follow any of the recommendations that are offered: "*Services of the UGR*", "*Studies of the UGR*", and "*Activities of the UGR*".

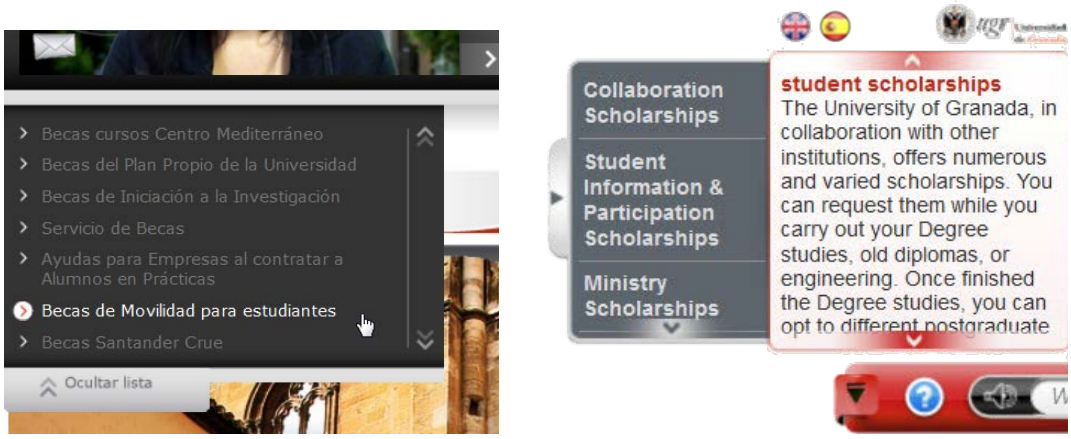
As well as the recommendation list, the virtual assistant can also show a sub-

(a) Initial version (top-sliding, 2D model, Spanish)

(b) New version (overlay, 3D model, multilingual)

Figure 4.10: The user interface

ject list like the ones appearing in Figure 4.11. Although they may seem the same thing, there is a little hint that distinguishes them. Unlike recommendations, which usually provide different alternatives to continue the conversation and are updated every time the user asks a new question, subject lists remain accessible throughout the dialog so that a user might consult several alternatives

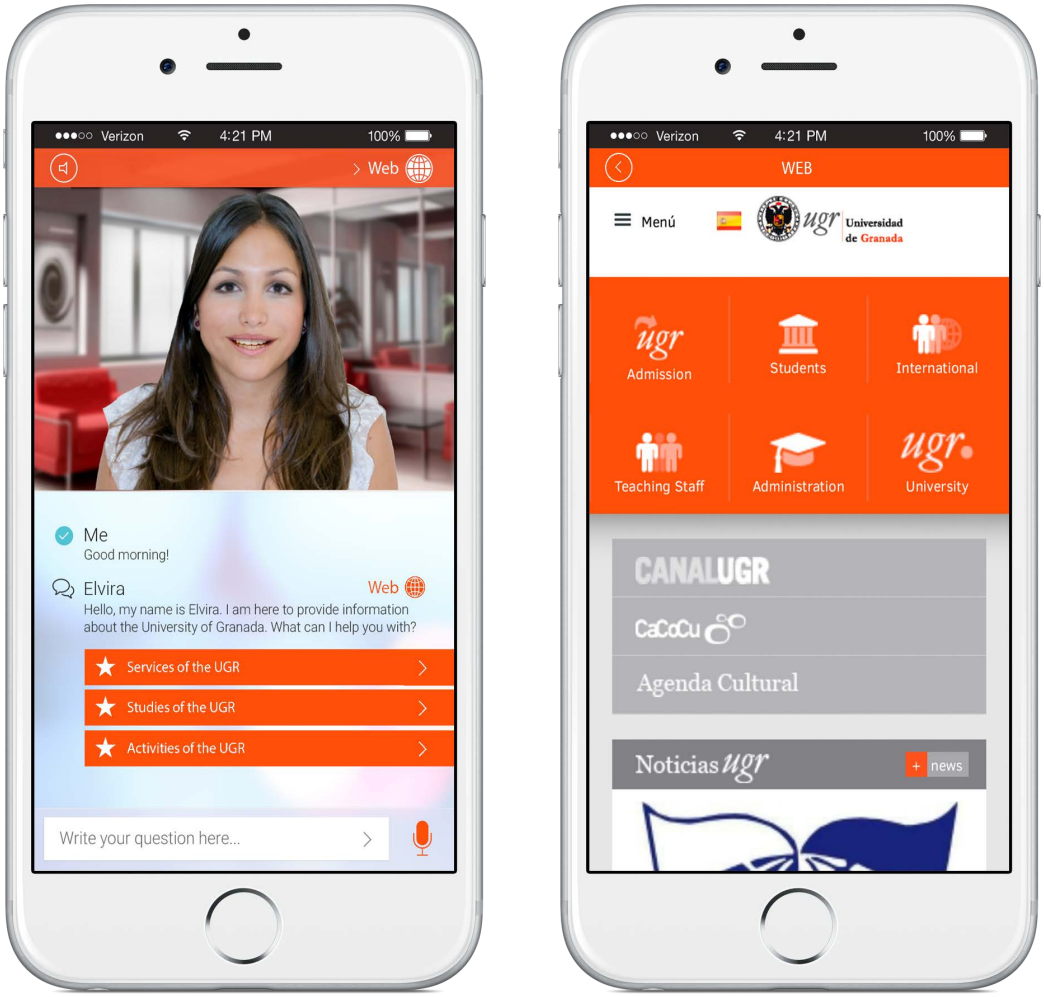


(a) Top-sliding version (b) Overlay version

Figure 4.11: The subject list

without having to scroll up the conversation record. For example, if a user asked Elvira about the "*student scholarships*", she would unfold a subject list with all the different kinds of student scholarships, as can be seen in Figure 4.11. In this way, the user might click and receive detailed information and related topics (i.e. recommendations) about several of these alternatives. Anyway, since the difference between both types of list is very subtle and sometimes it is difficult for users to see it, the interface can also be configured to always show subjects as if they were recommendations, that is, just below the answer to each question.

On the other hand, Figure 4.12 shows the user interface for smartphones. It is made of a main view and a web view. The main view shows the avatar of the virtual assistant (implemented natively using the C language to make it compatible with mobile devices), the conversation record with all questions and answers, and a text input together with a microphone button to type or ask a new question. The web view allows users to get additional information from the web pages associated to the different answers. This view can be accessed using the "Web" buttons that appear in the main view. We can also say that initially we made a third view to the left of the main view to include the subject list. However, it was removed to improve the user experience, so that both subjects and recommendations appear below each answer.



(a) Main view (center) (b) Web view (right)

Figure 4.12: The user interface for smartphones

4.5 Results

In this section, we will analyze the reliability, effectiveness and efficiency of our system. For this purpose, we will consider Elvira [1], the virtual assistant that provides information about the University of Granada [2] (Spain). The kind of information that she is able to supply is very wide: offices, services, studies, access, registration, faculties and schools, courses, activities, regulations, forms, schol-

[1] http://tueris.ugr.es/elvira/?lang=en
[2] http://www.ugr.es/

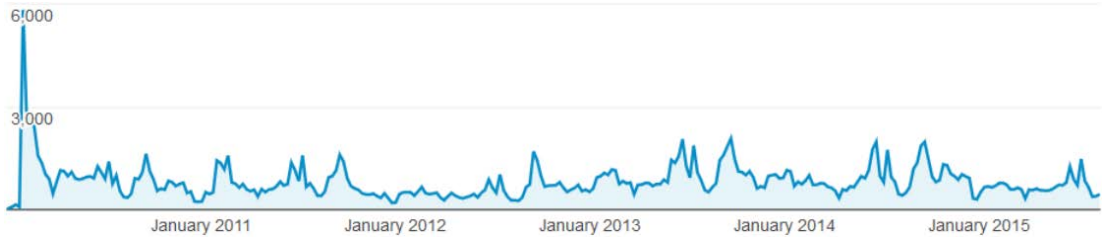


Figure 4.13: Weekly sessions (Google Analytics)

arships, accommodation, projects, and so on. In all, more than 2,500 different aspects related to this university.

At this moment, all this information is publicly offered just in Spanish, but, as we can see in Figures 4.10b and 4.11b, a multilingual version including the English language has already been developed. It is expected that this new version will be available to the general public by the end of 2015.

4.5.1 Sessions

This service of virtual assistance of the University of Granada is accessible since February 1, 2010. During the first five years and a half of use, there have been over 240,000 visits of approximately 190,000 unique users. This means, on average, 118 sessions a day, 823 a week, and 3,572 a month. As can be seen in Figure 4.13, there are some interesting patterns in this time series. During the first week, Elvira attended around 6,000 visitors, maybe due to the novelty of the service. The number of sessions is considerably lower in August than in July and September. These valleys are mainly due to the fact that August is the most traditional holidays month in Spain and the university is almost completely closed during that month. In contrast, July and September are very active months that include the Spanish university access tests, the enrollment, the start of the new course, etcetera, and these are some of the contents in most demand, as we will see in Section 4.5.5. On the other hand, the number of visits on working days usually doubles or triples the visits on weekends.

New visitors represent 78.06% of sessions. This counts the number of users whose session was marked as first-time access to Elvira. The remaining 21.94%

corresponds to returning visitors, that is, users whose session was marked as second or more time. These numbers must be caught carefully since they are based on the use of cookies to track user's behavior, and these are specific to the browser and device that have been used to access the system. In this way, if a user accessed Elvira for the first time via her desktop computer and then using her tablet, she would be considered a new visitor on both occasions. This would also happen if she accessed from the same device but clearing her cookies in between.

During these five years and a half, users have asked more than 530,000 questions and followed over 450,000 recommendations offered by Elvira. This makes a total of almost 1,000,000 answers.

The average number of questions asked by session is 2.0, and the number of recommendations followed is 1.9. Conversations are usually short, and each user receives 3.9 answers. Questions have an average length of 3.9 words, and the session duration is 3:17 minutes.

On the other hand, the bounce rate, that is, the percentage of sessions in which the user left the site without interacting with the virtual assistant, is 5.54%.

4.5.2 Devices

Regarding the device category, 86.65% of sessions come from desktop computers, 10.53% from mobile devices, and 2.82% from tablets. The preferred desktop operating system is Windows (79.63%), followed by Macintosh (4.65%), and Linux (2.40%). On the other hand, Figure 4.14 shows how the access from portable devices has increased significantly in the last few years. This reveals the necessity of counting with a native or hybrid app adapted to this type of devices, like the one shown in Figure 4.12. The favorite mobile operating systems are Android (10.33% of total devices, 77.39% of mobile devices), iOS (2.39% of total, 17.89% of mobile), and Windows Phone (0.13% of total, 0.97% of mobile).

4.5.3 Sources

Just 11.80% of traffic is direct, whereas 87.37% is referral. Most users come from the main web page of the university, which provides 67.42% of visits, and all its

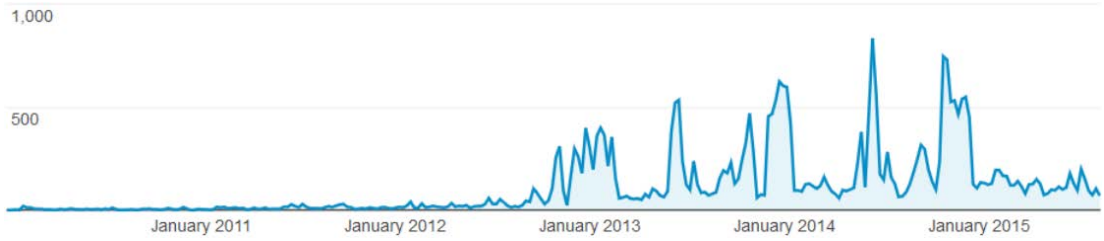


Figure 4.14: Weekly sessions from mobile devices (Google Analytics)

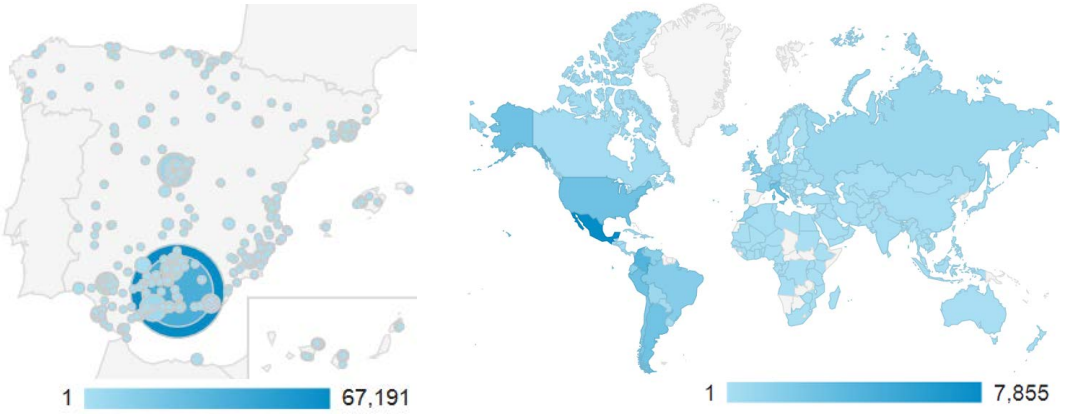
subdomains (i.e. offices, services...), which contribute an 18.93%. Finally, only 0.83% of traffic is organic (basically Google, with a 0.81%), and most frequent keywords are "*virtual assistant*", "*elvira ugr*", "*ugr elvira*", "*assistant*", "*virtual assistant ugr*", and "*elvira university of granada*".

With regard to the traffic from social networks, it is only 0.34% of total, being Facebook (0.18%), Twitter (0.08%), and Tuenti (0.06%) —the "Spanish Facebook" aimed at teenagers and young people— the three main contributors.

4.5.4 Locations

Visitors come from 167 different countries distributed all over the world, as Figure 4.15b shows. Visits from Spain have been filtered out to make the scale smaller and can see the differences between all the other countries. The top ten are Spain (80.78%), Mexico (3.23%), Colombia (1.69%), Peru (1.14%), United States (1.13%), Argentina (1.12%), Italy (1.11%), Chile (0.82%), Brazil (0.70%), and United Kingdom (0.62%). On the other hand, Figure 4.15a shows how visitors are also quite spread out in Spain, although many of them are situated in Granada and the surrounding cities, since it is where most of the people to whom Elvira is aimed live. Thus, the top ten cities are Granada (43.77%), Madrid (6.55%), Malaga (3.62%), Seville (3.01%), Jaen (2.24%), Almeria (2.02%), Barcelona (1.53%), Cordoba (1.19%), Valencia (0.91%), and Murcia (0.86%).

Although the English version of Elvira is not available to the general public yet, 13.13% of total visits correspond to non-Spanish speaking users. The most frequent foreign languages are English (60.02%), Italian (8.38%), French (7.57%), Portuguese (5.83%), and German (3.89%). We have also detected that some of



(a) Spain (b) Other countries

Figure 4.15: Sessions by country (Google Analytics)

these visitors end up using a translation service such as Google Translate to ask questions and read the provided answers in their own languages. This makes evident the importance of having a virtual assistant with multilingual capabilities.

4.5.5 Contents

Users have asked Elvira over 530,000 questions and have followed more than 450,000 recommendations offered by her.

Rude questions represent 9.35%, whereas the information units (IUs) in most demand have been Studies (5.61%), Enrollment (5.52%), Pre-enrollment (3.51%), Services (2.72%), Access to the university (1.73%), Cut-off grades (1.69%), Scholarships (1.60%), Student scholarships (1.43%), Academic calendar (1.35%), and Degrees (1.34%). According to the definition of the different types of IU given in Section 4.3.1, in this top ten list we can find one general subject (Scholarships), two subject lists (Access to the university and Student scholarships), and seven specific subjects. Moreover, as we can see in Figure 4.16, the 80.05% of the answers are concentrated on just the 8.55% of IUs. All this information advises the University of Granada to, instead of focusing excessively on IUs that are rarely asked, put most of the efforts dedicated to the maintenance of Elvira into improving and detailing these top contents, because sometimes they happen to be too general with regard to the actual questions asked by users.

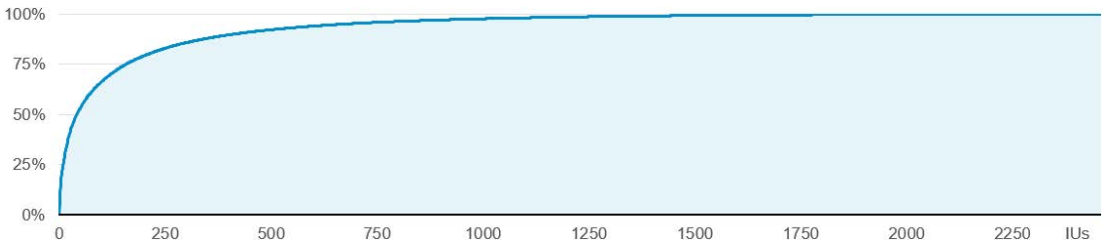


Figure 4.16: Answer's concentration on information units

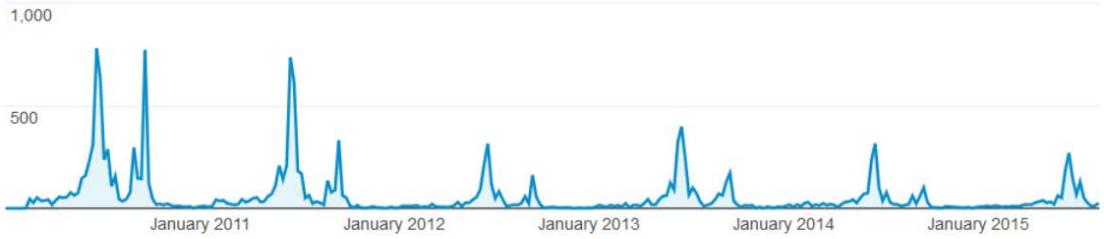


Figure 4.17: Questions about the pre-enrollment process (Google Analytics)

If we analyze the distribution of IUs considering the month of the year in which they were asked, we can find some interesting behaviors. For example, Figure 4.17 shows the distribution of the questions about the Pre-enrollment process. As we can see, most of them concentrate in June and September, the two months in which the pre-enrollment in the university takes place. The same happens with the questions about the Entry exams, the Enrollment itself, which concentrate in July and September, or the Academic calendar, which appear especially around September. In contrast, other topics such as Studies are much more scattered all over the year.

On the other hand, two of the top ten IUs (Studies and Services) are recommendations offered by Elvira at the beginning of the conversation. The third recommendation, Activities, occupies position 12. This shows the relevance of the information included in the Welcome IU. As we explained in Section 4.3.5, restrictions may be used to promote or provide punctual information about certain aspects of the university (e.g. official registration periods, cultural events, latest news...), and some of the best places to do it are the greeting messages. In fact, the 30.16% of IUs have been used by Elvira in less than six occasions during

these five years and a half.

4.5.6 Reliability

Regarding the reliability of the system, we have analyzed the different types of errors taking the domain of questions into account. The 65.59% of questions are related to the University of Granada, whereas the other 34.41% are out-of-domain questions. This means that the human appearance of Elvira encourages users to interact with her as if she was a real assistant, asking personal questions about her lifestyle, her hobbies, or any other topic for which she was not designed.

We have considered two types of errors. On the one hand, questions understanding errors made by Elvira, that is, how many times she was not able to recognize anything from the user's question and she was forced to use the search engine of the university. On the other hand, misunderstanding errors, which include the number of times that Elvira answered using a wrong information unit.

Understanding errors represent the 21.45% of total questions. At first sight this rate looks huge, but actually it does not mean a serious problem. Only the 10.68% of these errors are in-domain, that is, related to the University of Granada. This is equivalent to the 3.49% of in-domain questions, or the 2.29% if we also consider out-of-domain questions. These errors are typically due to topics that were not considered when the knowledge base was made (e.g. "*the mutual insurance company*"), and the fact that some specific ways of asking about certain topics are missing from the token files. The other 89.32% of understanding errors belongs to out-of-domain questions, that is, too general questions, unimportant dialog, or questions out of the scope of this virtual assistant (e.g. "*education*", "*tell me a joke*", or "*what's a molecule*"). Nevertheless, if we wanted to reduce the number of out-of-domain understanding errors, we could follow a multi-agent approach like the one described in Chapter 5, which would allow us to add knowledge from external data sources (e.g. Wikipedia) in an easy way.

On the other hand, misunderstanding errors represent the 2.24% of questions. These errors are caused by the lack of some expressions in the token files, or the conflict among the different recognition rules. To solve them, sometimes more tokens must be added, and other times the rules and the priorities of the

recognition algorithm must be adjusted.

In this way, the overall recognition rate is equal to 94.39% for in-domain questions, or 76.31% if we also consider out-of-domain questions, although we must make clear that answering the latter is not the main aim of Elvira.

Additionally, we have detected that sometimes the knowledge demanded by users is much more specific than the one that was included in the virtual assistant initially. To solve this type of problem, it is necessary to perform a continuous maintenance that can detect these lacks of information. In this way, the system can be completed little by little according to the interests of users, since dealing with all topics at a great detail level from the start could be impossible in a domain like this due to the vast amount of existing information.

Finally, another estimated measure of the quality of the recognition could be the declaration of gratitude. Around 3,000 people (1.27%) explicitly thanked Elvira for having helped them. However, after analyzing some of these sessions manually, we have detected that a minor number were ironic because those users did not find the information that they were looking for (it was too specific, or it had not been included into the system). Even so, if we skipped all the sessions that received a question understanding error at any point of the dialog, which would give us a quite (maybe too much) conservative rate, we would still have more than 2,300 thanks (0.97%).

4.6 Conclusion and future work

In this chapter we have studied the problem of information overload in websites. As information grows, it is more and more difficult to organize it in an appropriate way so that users can easily find it. In addition, sometimes the information about a certain topic is scattered throughout the website, so it is even more difficult to find. For this reason, we have reviewed some of the existing systems for improving web navigation. As we have seen, they presented some disadvantages for being applied to a domain like the one of the University of Granada, which entails some particular challenges: great amount of information spread over different independent subdomains; various user groups with different interests; dynamic information which varies throughout the time; presence of external resources;

possibility of informing in several languages or at different detail levels, and so on.

In order to solve these problems and better support users in navigation, we have presented a methodology and a framework for designing closed-domain virtual assistants which allow accessing a great amount of information in an immediate and precise way. These have been used for the development of Elvira, the first virtual assistant of a Spanish university, which has answered more than 500,000 questions during its first five years and a half of use.

Despite the good results obtained throughout this time, a lot of work must be done to improve some aspects. For example, one of the bottlenecks in the development of these systems is token generation. There are many different expressions that people can use to refer to each concept, and these must be specified to make the system able to answer questions about them. Making all these expressions completely by hand is a quite difficult task, especially when there are many concepts that are in conflict with each other. However, the use of synonyms or the (semi) automatic generation of tokens from a corpus of questions labeled with the right answer can be of great help. Therefore, although we have developed some tools which make the token generation and the answer validation processes easier, we think that it is very important to keep working on this way, since it has direct repercussions on the reliability and quality of the final result.

On the other hand, it would be necessary to work on website monitoring for detecting changes in answers and discovering new contents automatically. This would help us to synchronize the knowledge of the virtual assistant with the information of the website. In addition, we could identify which contents of the virtual assistant are too general regarding the actual questions asked by users, and decide whether to go into more depth.

Finally, the recommendations offered by the assistant are static at this moment. They are adapted to the conversation context, but they do not take account of the behavior of each particular user or the moment when they are offered. We think that the use of a more dynamic and collaborative approach could improve the efficiency of recommendations, which, as we have seen, are followed by users frequently. For example, we could analyze the evolution of the number of answers about a certain information unit with regard to previous year in order to predict if

it is going to be popular during the next few days and it should be recommended at the beginning of the conversation.

This page intentionally left blank

5

Accessing heterogeneous data sources

In many of the problems that can be found nowadays, information is scattered across different heterogeneous data sources. Most of the natural language interfaces just focus on a very specific part of the problem (e.g. an interface to a relational database, or an interface to an ontology). However, from the point of view of users, it does not matter where the information is stored, they just want to get the knowledge in an integrated, transparent, efficient, effective, and pleasant way. To solve this problem, this chapter proposes a generic multi-agent conversational architecture that follows the divide and conquer philosophy and considers two different types of agents. Expert agents are specialized in accessing different knowledge sources, and decision agents coordinate them to provide a coherent final answer to the user. This architecture has been used to design and implement SmartSeller, a specific system which includes a Virtual Assistant to answer general questions and a Bookseller to query a book database. A deep analysis regarding other relevant systems has demonstrated that our proposal

provides several improvements at some key features presented along the chapter.

5.1 Introduction

As many studies reveal (Chai et al., 2001b; Chai et al., 2001c; Kaufmann and Bernstein, 2010; Zhou et al., 2012), there is a clear preference of users for full natural language query interfaces rather than keywords, formal query languages, or menu driven interaction. In addition, the interest of users in a particular site decreases exponentially with the increase in the number of mouse clicks (Huberman et al., 1998). This fact is emphasized even more if we talk about mobile devices, where traditional input interfaces are very limited. Natural language systems are able to improve the perceived usefulness, ease-of-use, and efficiency, which in turn account for positive attitude and intention to use those systems.

From an economical point of view, the increase of e-commerce spending supposes a great opportunity for natural language interfaces. According to comScore [1] (one of the most important Internet marketing research companies), Q1 2014 saw desktop-based U.S. retail e-commerce spending rise 12% year-over-year to $56.1 billion, marking the eighteenth consecutive quarter of positive year-over-year growth. M-commerce spending on smartphones and tablets added $7.3 billion for the quarter, up 23% vs. year ago, for a digital commerce spending total of $63.4 billion in the first quarter. In addition, some events like Alibaba's [2] Singles' Day sales confirm that m-commerce is more and more important. During that day, the world's biggest online retail sales day [3], sales exceeded predictions at $9.3 billion, shipping 278 million orders, 43% of which were placed on mobile devices. This change in consumer behavior reflects the necessity of defining more intelligent ways of interacting with websites and product databases rather than traditional keyword search.

This chapter focuses on the problem of accessing heterogeneous data sources using natural language dialog. As we will see in Section 5.2, most of the natural

[1] https://www.comscore.com/Insights/Press-Releases/2014/5/comScore-Reports-56-1-Billion-in-Q1-2014-Desktop-Based-US-Retail-ECommerce-Spending-Up-12-Percent-vs-Year-Ago

[2] http://www.alibaba.com/

[3] http://www.bbc.com/news/business-29999289

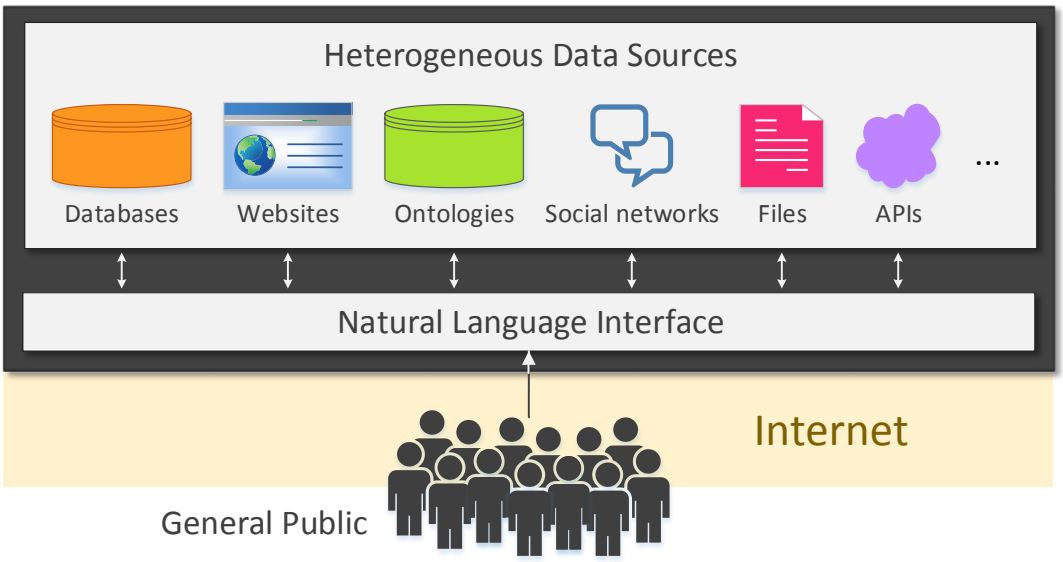


Figure 5.1: Natural Language Interface for general public

language interfaces (NLIs) that can be found in literature are usually oriented to solve very specific problems in which the knowledge is stored in just one type of source (e.g. NLIs to relational databases, NLIs to ontologies...). However, the reality is usually quite different.

As can be seen in Figure 5.1, when users interact with an NLI, they have a single objective in mind: getting the information they want at that moment. From their point of view, it does not matter if it is an NLI to a database, an ontology, or an XML file, nor if it is a virtual assistant that can instruct them in using a website or answer general questions. What is really important is that, if we want these systems to be applied to many different fields and be extensively used by the general public and not only by expert users or database administrators, they must put all that heterogeneous information at the service of users in an integrated, transparent, efficient, effective, and pleasant way.

As we will see in Section 5.3, the architectures used by this type of systems present some advantages and certain disadvantages. And although many of these systems have been more or less usable in practice for different application domains, they present several problems. Many of them are not interactive or their

interactivity is very limited. However, users often do not know where to find the information, or their requests are vague and they need to complete them little by little with additional information, which could be provided in a dialog. For this reason, the lack of memory and a dialog management module to keep track of the conversation and use the context in an appropriate way is a big problem. Other issues regarding these systems are that most are not ready to work with heterogeneous data sources at the same time and in a transparent way. Moreover, the majority cannot handle fuzzy concepts such as "*cheap*" or "*recent*", or temporal queries involving relative dates, which is essential. On the other hand, they do not use 3D characters that can emotionally engage in conversation with users, making the interaction process friendlier. Finally, sometimes they are simple prototypes that are far from being a powerful and easy to use system that can be used in a real environment with a large number of users of any type.

Our proposal to address these problems and improve what other systems provide is to use a multi-agent approach that allows applying a divide and conquer philosophy. In this way, a set of expert agents specialized in concrete domains facilitates the access to different knowledge sources, and a series of decision agents interacts with them to coordinate them and provide a unique final answer to the user. This proposal will be analyzed in depth in Section 5.4, and it will include the design and development of a specific conversational system for a bookstore. This system will be used in Section 5.5 to make a comparative analysis and present some implementation and usage details in order to confirm that our proposal provides several improvements in the aforementioned features. Finally, Section 5.6 will include some general conclusions and will present some directions for future work.

5.2 Related work

The problem of Natural Language Interfaces (NLIs) started more than four decades ago. NLI systems are divided into three main categories, depending on how knowledge is structured:

- NLIs to structured data: usually closed-domain, they can work either on

relational databases translating the query to SQL (the so called Natural Language Interfaces to Databases, or NLIDBs), or on ontologies translating the query to SPARQL, for example.

- NLIs to non-structured or semi-structured data: usually open-domain, they process huge amounts of documents to find the answer to a question.

- interactive NLIs: used in dialog systems, they have memory to remember previous questions.

In this chapter we will focus on NLIs to structured data and interactive NLIs. In order to be able to better understand the benefits of our proposal, we will show the main features of some NLIDB systems that can be found in literature, from classic implementations to state-of-the-art prototypes.

Since their appearance at the end of the sixties, many different systems have been proposed. In general, during the first years, all the systems were domain specific. They were tied to a particular database and they were difficult to port. They used to be based on pattern matching techniques and syntactic trees. During the seventies and eighties, this kind of systems were improved with the use of new techniques such as semantic grammars or intermediate representation languages, which were independent from the database. However, it was not until the end of that period that some commercial systems started to appear (Sijtsma and Zweekhorst, 1993) (e.g. Intellect (Artificial Intelligence Corporation, 1982), Natural Language, Q&A, LanguageAccess by IBM, English Query by Microsoft, or English Wizard by Linguistic Technology Corporation). In spite of their initial popularity, these systems gradually disappeared due to several problems: bad performance, significant effort required to develop specialized systems for individual databases that could not be easily adapted to work with different domains, language coverage not obvious to users, ambiguity of natural language, and so on. Nevertheless, after some years in the oblivion, they started to live a second youth with the arrival of the new century, and nowadays they are mature enough

as Apple's Siri [1], Google Now [2], Microsoft Cortana [3], or Amazon Echo [4] show, although these systems are much more complex than a simple NLIDB.

We will talk more about the different types of architectures of these systems in Section 5.3. But first, we will briefly describe some specific examples. Although in Section 5.5 we will make a thorough analysis considering some of these systems, more details on any of them can be found in the corresponding paper.

REL (Thompson et al., 1969), initially written in assembler code, provided a core language and allowed users to make interactive extensions to the system in order to adapt it to a particular database. It used a formal grammar expressed as a set of context-free and general rewrite rules with semantic transformations attached to each rule. Answers were obtained from a built-in database.

CONVERSE (Kellogg et al., 1971) performed a surface and deep-structure syntactic analysis, and used a network of concepts to construct semantic interpretations formalized as computable procedures which were evaluated by a data management system.

LUNAR (Woods et al., 1972) is one of the most well known NLIDB systems. It presented a natural language interface to a database containing chemical analyses of Apollo-11 moon rocks. LUNAR used an Augmented Transition Network (ATN) parser, Woods' procedural semantics, and two databases for chemical analysis and literature references. Although it was not broadly used, its performance was impressive for that time, with an accuracy of 78%, or 90% if dictionary errors were corrected.

RENDEZVOUS (Codd, 1974) was intended to engage users in free (relatively unrestricted) dialogs to help formulate queries over relational databases. It placed special emphasis on query paraphrasing and engaging users in clarification dialogs. The objective was to make sure that the intended meaning of the user's question had been correctly captured before routing the formal query to the relational DBMS.

TORUS (Mylopoulos et al., 1976) was a research oriented system that used a semantic network to store knowledge about a database of student files. That

[1] https://www.apple.com/ios/siri/
[2] https://www.google.com/landing/now/
[3] http://www.windowsphone.com/en-US/how-to/wp8/cortana/
[4] http://www.amazon.com/oc/echo

knowledge was used to find the meaning of each input statement, to decide what action to take with respect to the database, and to select information that had to be output in response to the input statement.

PHLIQA1 (Philips question answering system) (Scha, 1977; Bronnenberg et al., 1980) was an experimental system developed at Philips Research Laboratories for answering isolated English questions about a restricted subject domain. It used a syntactic parser which run as a separate pass from the semantic understanding passes, and it could interface to a variety of database structures and DBMSs.

LADDER (Hendrix et al., 1978) was designed to access information about US Navy ships. It could be used with large distributed databases and different database management systems. It used semantic grammars (a different grammar had to be defined for each application), parsing the user's question to build a semantic tree, which was then mapped to a database query. It also included facilities such as spelling correction and elliptic reasoning.

ROBOT (Harris, 1978) was one of the few systems commercially available. It was the basis for other systems such as Online English (Cullinane Corporation, 1980) and Intellect (Artificial Intelligence Corporation, 1982). The system used an extracted version of the database for lexical data to assist an ATN (Augmented Transition Network) parser.

PLANES (Programmed LANguage-based Enquiry System) (Waltz, 1978) was able to answer questions using a large relational database of aircraft flight and maintenance data. It carried out clarifying dialogs with the user and was able to answer vague or poorly defined questions. It used an ATN based parser and a semantic case frame analysis to understand questions.

EUFID (End-User Friendly Interface to Data management) (Templeton, 1979; Templeton, 1983) was a modular and table driven natural language front-end that could be interfaced to different applications and data management systems. It allowed including sloppy syntax and misspellings. The tables contained a data management system view of the database, a semantic/syntactic view of the application, and a mapping from the second to the first.

TQA (Transformational Question Answering) (Damerau, 1981), originally named REQUEST, was made for the Planning Department of the City of White

Plains, New York. It used several transformational and context-free parsers and rules, and a semantic interpreter with semantic rules.

CHAT-80 (Warren and Pereira, 1982) is another of the most popular classical systems. It transformed English questions about world geography into Prolog expressions, which were evaluated against the Prolog database. It used a small set of English language vocabulary, enough for querying the database. It formed the basis of many other systems such as MASQUE/SQL (Androutsopoulos et al., 1993) or BusTUC (Amble, 2000).

ASK (Thompson and Thompson, 1983; Thompson and Thompson, 1985) allowed users to teach the system new words and concepts during the interaction. It was a complete information management system which interacted with multiple external databases, electronic mail programs and other computer applications. In this way, user's queries generated requests to the appropriate underlying systems.

DATALOG (Hafner, 1984) was based on a cascaded ATN grammar. It provided separate representation schemes for linguistic knowledge, general world knowledge, and application domain knowledge to achieve a high degree of portability and extendibility.

LDC (Layered Domain Class) (Ballard et al., 1984) provided retrieval capabilities for medium-sized office domains stored as text files, as opposed to more restrictive database structures. It could also be extended into new domains without the intervention of the designer of the system, and without sacrificing the depth or reliability of the interface.

TELI (Transportable English-Language Interface) (Ballard and Stumberger, 1986) answered English questions about tabular data files. It allowed users to define, examine, and modify the definitions of any domain-specific words or phrases known to the system. It also enabled the design of customization modules that were largely independent of the syntactic and retrieval components.

TEAM (Grosz et al., 1987) focused on the design of portable systems which could be easily configured by database administrators with no knowledge of NLIDBs, as opposite to the high configuration costs of other systems. A database expert engaged in an acquisition dialog with TEAM to supply the information needed to adapt the system to a new database or to expand its capabilities in answering questions.

JANUS (Hinrichs, 1988) used multiple underlying systems (databases, expert systems, graphics devices, and so on) to evaluate natural language requests. In addition, it was one of the few systems which supported temporal questions.

MASQUE/SQL (Androutsopoulos et al., 1993) was a modified version of MASQUE (Modular Answering System for QUeries in English) (Auxerre and Inder, 1986), a portable natural language front-end for Prolog databases. It could be used as a front-end to any commercial database supporting SQL, although it had to be configured for each different domain and database.

Edite (Reis et al., 1997) was a multilingual (Portuguese, French, English and Spanish) NLIDB for a database with 53,000 tourism resources. It used an intermediate representation language to express the meaning of the sentence in terms of high-level concepts, independent of the database structure, before the final translation to a SQL query.

JUPITER (Zue et al., 1997; Zue et al., 2000) was a conversational interface that allowed users to obtain weather forecast information for over 500 cities worldwide over the telephone using spoken dialog. Like many other systems from the Massachusetts Institute of Technology (e.g. MERCURY for flight reservations (Seneff et al., 1999; Seneff and Polifroni, 2000; Seneff, 2002), ORION for off-line task delegation (Seneff et al., 2000), PEGASUS for flight arrival and departure information (Zue et al., 1994), and VOYAGER for city traffic and landmark information (Zue et al., 1992)), JUPITER made use of the GALAXY (Goddeau et al., 1994) conversational system architecture. It had a separate geography table organized hierarchically that provided a means of summarizing results that were too lengthy to present fully. An end-to-end Japanese version was developed (MOKUSEI (Nakano et al., 2001)).

MERCURY (Seneff et al., 1999; Seneff and Polifroni, 2000; Seneff, 2002) was an over-the-telephone NLI built within the GALAXY-II (Seneff et al., 1998) architecture. It provided flight schedule information and pricing potential itineraries involving the 150 busiest airports worldwide. This domain entailed additional challenges such as time zone differences or geographical proximity of airports. The system included a fuzzy matching heuristic, and users were permitted to specify and update constraints at any point in time.

CoBase (Zhang et al., 1999) used a semantic graph which modeled the objects in the database and user-defined relationships, and could be semi-automatically generated from a database schema. Queries were formulated by finding and ranking a set of paths in the semantic graph which encompassed the user input. An incremental approach allowed formulating complex queries through a series of simple queries.

BusTUC (Amble, 2000) was a bilingual route advisor for the public bus transport in Trondheim, Norway. It automatically detected which language was most probable by counting the number of unknown words related to each language. The system used a dictionary, a lexical processor, a grammar, a parser, a semantic knowledge base, a routing logic system, and a route database. It relied on an internal language independent logic representation.

Nelken and Francez (2000) presented a controlled natural language interface to temporal database systems which focused on the role of temporal preposition phrases rather than the more traditional focus on tense and aspect. The user had to learn the syntactic restrictions of the system to make sure she kept in the controlled natural language subset. The questions were translated into SQL/Temporal, a temporal database query language.

gNarLI (Shankar and Yung, 2000) was a rule-based pattern matcher written in Perl. Each domain (e.g. movies, or classes at Harvard) had different rules that could be matched to a portion of the input question by a regular expression pattern and had some bearing on the SQL query to be generated. It was not suited to handle huge statistical datasets (e.g. NBA statistics) with many columns, calculate averages and sums, compare, and sort in different ways.

EQUIsearch (Silverman et al., 2001) used restricted natural language to help users search product catalogs in large-scale market exchanges. They identified several challenges such as handling incomplete and inconsistent product descriptions, or the difficulty of matching the buyer's search terms to the wording in the descriptive fields. The recognizer was based on objects and attribute-value pairs, and three custom domain specific dictionaries were used for spelling, stripping and synonyms.

HappyAssistant (and the later enhanced versions Natural Language Shopping Assistant –NLSA– and Natural Language Assistant –NLA–) (Budzikowska et

al., 2001; Chai et al., 2001b; Chai et al., 2001c; Chai et al., 2001a; Chai et al., 2002) was an interactive NLI that helped users access e-commerce sites to find relevant information about products and services. It used a rule system with weights and ranks to implement business rules that were able to display the web page that satisfied the user's requests or initiate a dialog to ask for additional information. Moreover, when a product was recommended to the user, it generated an explanation automatically. The NLA version included a statistical parsing that allowed scaling to multiple languages and geographies with minimal reconfiguration.

Zhang et al. (2002) proposed a multilingual NLIDB that used case based reasoning (CBR) (Aamodt and Plaza, 1994) for a stock market information retrieval system. It avoided constructing parsers for all supporting natural languages by storing every query pattern and its solution as a case into a case base. The user's sentence was syntactically compared to the cases in the case base, and the solution of the most similar case was reused to query the database. Each case was represented as an XML document.

Dittenbach et al. (2003) presented an interface for querying the largest Austrian tourism platform Tiscover in German and English. The language of the query was automatically detected using an n-gram-based text classification approach. The user's query concepts were translated into parameterized SQL fragments. An ontology was used to specify the concepts that were relevant in the application domain and to describe linguistic relationships like synonymy.

PRECISE (Popescu et al., 2003) introduced the idea of semantically tractable sentences, which could be translated into a unique semantic interpretation by analyzing some lexicons and semantic constraints. It matched keywords in a sentence to the corresponding database elements. First, it narrowed the possibilities using the max-flow algorithm, and then, it analyzed the syntactic structure of the sentence. The evaluation was performed on two databases: ATIS (questions about flights) and GeoQuery (U.S. Geography). PRECISE was able to achieve high accuracy in semantically tractable questions, although it compensated for the gain in accuracy at the cost of recall.

Semeraro et al. (2003) presented a proactive agent-based interface based on the COGITO (Intelligent E-Commerce with Guiding Agents Based on Personalized

Interaction Tools) (Thiel et al., 2003) architecture which was able to engage in a goal-directed conversation with the user. It was represented by a 2D facial model. It made a user profile and it combined content-based retrieval with collaborative filtering to recommend new products. Queries were modified by adding search terms taken from three different sources: product thesaurus, user profiles and usage patterns.

EXACT (Yates et al., 2003) was an NLI to household appliances (NLIA) based on PRECISE (Popescu et al., 2003) and the Blackbox planner (Kautz and Selman, 1998). It took an input sentence, converted it into a set of possible SQL statements using PRECISE, translated those into a set of PDDL goals, and looked for a plan (a sequence of appliance commands) to satisfy each goal.

SAMIR (Scenographic Agents Mimic Intelligent Reasoning) (Zambetta et al., 2003; Abbattista et al., 2004; Zambetta and Abbattista, 2005) was a slightly modified version of ALICE (an open source chatterbot developed by the ALICE AI Foundation and based on the AIML language) which provided the user with dialogging capabilities and used a 3D face to convey expressions. It defined some AIML categories to let users navigate in a bookstore, considering a set of seven fields: title, author, publisher, publication date, subject, ISBN code, and keyword. However, AIML categories did not fit well for user requests that exhibited a high level of ambiguity. Regarding the behavior of SAMIR, it was controlled by a set of random generated if-then rules and its capability was very low.

Ginseng (Guided Input Natural language Search ENGine) (Bernstein et al., 2005) was a quasi natural language guided query interface to the Semantic Web. It relied on a simple grammar which was dynamically extended by the structure of an ontology. Based on this grammar, the system offered pop-up boxes which suggested how to complete the queries and prevented those unacceptable by the grammar. Once completed, queries were translated into RDQL, a semantic web query language.

NaLIX (Natural Language Interface for an XML Database) (Li et al., 2005) translated arbitrary English sentences (which could include aggregation, nesting, and value joins) into XQuery expressions that could be evaluated against an XML database. The transformation process was done in three steps: generating a parse tree, validating the parse tree, and translating the parse tree to an XQuery

expression. It was tested on the XML database Timber.

STEP (Minock, 2005) was an NLIDB written in LISP that adopted a phrasal approach where an administrator coupled phrasal patterns to elementary expressions of tuple relational calculus. This 'phrasal lexicon' enabled the generation of natural language from tuple relational calculus and vice versa. This allowed engaging users in clarification dialogs when the parse of their queries was of questionable quality or was open to multiple interpretations.

Pazos Rangel et al. (2005) and González B. et al. (2007) presented a domain independent NLIDB that automatically generated the domain dictionary for query translation using semantic metadata of the database. The query was represented as a graph that was translated taking into account the parts of speech of its words. Prepositions and conjunctions were represented as operations using formal set theory which improved the results of translation in complex queries.

Stratica et al. (2005) proposed a template-based NLIDB system that was used in the CINDI virtual library. The input sentences were syntactically parsed using the context-free Link Parser (Sleator and Temperley, 1991) and semantically parsed using domain-specific templates. The system was composed of a preprocessor and a runtime module. The preprocessor built a conceptual knowledge base from the database schema using WordNet [1] (Miller et al., 1990; Miller, 1995; Fellbaum, 1998), which was used at runtime to semantically parse the input and create the corresponding SQL query.

WASP (Word Alignment-based Semantic Parsing) (Wong and Mooney, 2006) was designed to construct a complete, formal, symbolic, meaningful representation of a natural language sentence. It learned to build a semantic parser given a corpus of natural language sentences annotated with their correct formal query languages. The whole learning process was done using statistical machine translation techniques with minimal supervision, so it was not necessary to manually develop a grammar in different domains. WASP was evaluated on the GeoQuery domain and on a variety of natural languages (English, Spanish, Japanese and Turkish).

GINO (guided input natural language ontology editor) (Bernstein and Kaufmann, 2006) allowed users to edit and query ontologies in a language akin to

[1] http://wordnet.princeton.edu/

English. It used a small static grammar, which was dynamically extended with elements from the loaded ontologies. This allowed for easy adaptation to new ontologies.

Hallett (2006) presented a system that used a semantic graph to automatically generate query frames. It inferred the set of possible queries that could be applied to a given database, and generated a lexicon and grammar rules for expressing those queries. Then, users completed those frames through a list of search values. For that reason, it included neither syntactic parsing nor semantic interpretation generation.

Querix (Kaufmann et al., 2006) was a domain-independent NLI for the Semantic Web based on clarification dialogs. Rather than solving ambiguous natural language queries, it asked the user for clarification by showing a dialog box with several options. It used the Stanford parser (Klein and Manning, 2002) and a set of heuristic rules to translate the natural language queries into SPARQL.

Let's Go (Raux et al., 2006) was a telephone-based bus schedule information system used in Pittsburgh, USA. It was based on the RavenClaw dialog manager (Bohus and Rudnicky, 2003), which provided a set of domain independent dialog strategies for handling non-understandings. It tried to keep system requests as specific as possible since long prompts were not well received by users and were mostly ineffective.

Neva (Ahad et al., 2007) was a conversational agent for library information systems. The user was identified with the use of a Radio Frequency Identification (RFID) tag. It could be installed in a kiosk based information system. It used a local library database and a built-in Amazon web service for accessing book information.

Jusoh and Al-Fawareh (2007) presented a natural language interface for an online sales system based on several agents and a knowledge based system. It used a relatively small ontology and a methodology that was only tested on grammatically correct and complete sentences.

NLP-Reduce (Kaufmann et al., 2007) was a "naïve" domain-independent NLI for the semantic web. It was a simple approach which processed queries as bag of words. It employed a reduced set of techniques such as stemming and synonym

expansion using WordNet [1] (Miller et al., 1990; Miller, 1995; Fellbaum, 1998). It allowed using full natural language queries, sentence fragments, or just keywords. It identified subject-property-object triples that were joined and translated into SPARQL statements which were executed using Jena [2] and the Pellet Reasoner [3] (Sirin et al., 2007) to infer implicit triple statements.

Owda et al. (2007) proposed combining goal oriented conversational agents and knowledge trees. On the one hand, goal oriented conversational agents were used to disambiguate the user's queries and converse within a context. On the other hand, knowledge trees worked as a road map for the conversational agent dialog flow.

PANTO (Portable nAtural laNguage inTerface to Ontologies) (Wang et al., 2007) removed the need to learn the syntax of RDF or OWL, the formal query language, or the schema and vocabulary of ontologies. It used a triple-based data model as the intermediate representation to translate natural language queries into SPARQL.

ORAKEL (Cimiano et al., 2007b; Cimiano et al., 2007a; Cimiano et al., 2008) focused on minimizing the effort of adapting the system to a given domain. It used an ontology to guide the lexicon construction process and an inference engine to provide an answer, even if it was not explicitly contained in the knowledge base but can be inferred from it. For this reason, questions had to be translated into logical form.

Küçüktunç et al. (2007) proposed an NLI system for querying a video database. Queries were grouped as object-appearance, spatial, and similarity-based object trajectory by using POS tagging information. These were sent as Prolog facts to a query processing engine, which interacted with both a knowledge base and an object-relational database to respond to spatiotemporal, semantic, color, shape, and texture video queries.

DaNaLIX (Domain-adaptive Natural Language Interface for Querying XML) (Li et al., 2007) was an extension of NaLIX (Li et al., 2005). While retaining the portability of a purely generic system like NaLIX, DaNaLIX could exploit domain

[1] http://wordnet.princeton.edu/
[2] https://jena.apache.org/
[3] http://clarkparsia.com/pellet/

knowledge (whenever available) to improve its usability and query translation accuracy. This domain knowledge could be automatically obtained from the interactions between a user and the system.

MaNaLa (Mapping Natural Language) (Giordani, 2008) built syntactic propositional trees from natural language questions and relational trees from SQL queries. Similar trees were classified using a tree kernel, which computed the number of common substructures between two trees, generating a feature space. This classification was exploited to answer natural language questions, generating SQL codes that were ranked based on the number of matching substructures.

Huang et al. (2008) proposed a system including a probabilistic context free grammar. Since usually more than one grammar tree might be generated, it used an algorithm to calculate inside and outside probabilities and select the most probabilistic grammar tree to proceed translating the natural language query to SQL.

C-Phrase (CatchPhrase) (Minock et al., 2008; Minock, 2010) was an NLI system that could be configured using a web-based GUI reducing the necessary time and expertise. It modeled queries in an extended version of Codd's tuple calculus and used synchronous context-free grammars with lambda-expressions to represent semantic grammars. The given grammar rules might be used in the reverse direction to achieve paraphrases of logical queries. The result was automatically mapped to SQL. The system was evaluated on the GeoQuery 250 corpus.

NLION (Natural Language Interface for querying ONtologies) (Ramachandran and Krishnamurthi, 2009) transformed natural language questions into SPARQL queries. It used semantic relation tagging to recognize the meaning of the user query with respect to the target ontology.

Ruiz-Martínez et al. (2009) presented a method for querying ontologies which used natural language processing techniques for analyzing text fragments, ontologies for knowledge representation, and semantic web technologies for querying ontological knowledge bases. It was applied to the e-tourism domain.

FREyA (Feedback, Refinement and Extended vocabularY Aggregation) (Damljanovic et al., 2010; Damljanovic et al., 2012) was an interactive natural

language interface for querying ontologies and the Linked Open Data (LOD) [1]. It used the Stanford Parser (Klein and Manning, 2002), an ontology-based lookup, and usability methods such as feedback and clarification dialogs for training the system and improving its performance over time. It was tested on GeoQuery and the DBpedia (Auer et al., 2007; Bizer et al., 2009).

STK (Syntactic Tree Kernel) (Giordani and Moschitti, 2010) automatically learned a model based on lexical and syntactic description of the training examples, pairs of natural language questions and SQL queries. Such relational pairs were encoded in support vector machines by means of kernel functions applied to the syntactic trees of questions and queries.

Khamis and Shatnawi (2010) presented a semiautomatic framework for building domain specific ontologies that could be used to understand and interpret indirect user's queries and convert them into simple and semantic meaning free requests that could be directly processed by existing natural language applications. Data conceptual models were used to create the initial taxonomy of ontologies, which were enriched using WordNet [2] (Miller et al., 1990; Miller, 1995; Fellbaum, 1998) and refined by domain experts removing the unnecessary parts.

Lim and Lee (2010) proposed a natural language interface to web services described in OWL-S (World Wide Web Consortium, 2004). The user's query was parsed with the RASP system (Briscoe et al., 2006) and, using templates, it was translated into an abstract workflow which described the constituent tasks and their transitions in a composite service. The action and the object of each sentence block were determined based on its verb and nouns, and they were used to discover the appropriate service which was mapped to each task.

Kuchmann-Beauger and Aufaure (2011) built an NLI for a data warehouse based question answering system. It was composed of a string matching component performing keyword-based question answering, and a learning thesaurus automatically built from the data model and enriched through users' queries. This thesaurus was used to rewrite the queries.

SEM (Semantically Enriched Model) (Pazos R. et al., 2011) was a new modeling method for databases that included semantic information (besides the struc-

[1] http://linkeddata.org/
[2] http://wordnet.princeton.edu/

tural information of the database) in order to facilitate the translation process from natural language to SQL. It was based on the analysis of problems that occur in the corpora of three databases: ATIS, Northwind and Pubs.

Pixel (Allison, 2012) was a pilot chatbot that answered questions about the library of the University of Nebraska-Lincoln and its resources. It was built using Program-O [1], an open source PHP AIML interpreter. Chat logs revealed some interesting notes about its use: 34% of searches were about database subjects or look-ups for specific titles; 16% involved questions about services (e.g. "*How do I renew a book?*"); 15% were multi-step and undirected searches; ready reference represented 11%; 10% were about system problems (e.g. login authentication issues); simple answer questions were 6%; personal questions, 4%; and questions that involved non-library issues, and directional inquiries about where to find something, 4%. Complex questions were referred to librarians.

NaLIR (Li and Jagadish, 2014b; Li and Jagadish, 2014a) was a generic interactive NLI for querying relational databases. It used the Stanford Parser (Klein and Manning, 2002) and generated a SQL query that might include aggregation, nesting, and various types of joins. An interactive communicator explained how each query was processed and solved ambiguous interpretations generating multiple choice selection panels. It was applied to different domains: Microsoft Academic Search [2], Yahoo! Movies [3], and the DBLP bibliography (Ley, 2002; Ley, 2009) [4].

Regarding commercial systems, one of the most famous examples is Anna [5], IKEA's Automated Online Assistant, created by Artificial Solutions [6]. Like most conversational agents, she has a memory and is able to answer a quite broad set of general questions (e.g. personal information, or hobbies). But, if we focus on the IKEA-related knowledge, although she answers questions about products (e.g. "*I need a black table*"), opening hours, payment methods, or delivery charges successfully, she fails answering a little bit more complex questions like "*I need*

[1] http://www.program-o.com/
[2] http://academic.research.microsoft.com/
[3] http://movies.yahoo.com/
[4] http://dblp.uni-trier.de/
[5] http://www.ikea.com/us/en/
[6] http://www.artificial-solutions.com/

a cheap black table", "*I need a black table under $100*", or "*What are the best selling black tables of 2015?*". In both cases, Anna's answer is: "*We have many options for black table in our catalogue. You can click on your favorite on the webpage I have opened to you.*".

In this chapter we will not analyze other commercial systems, although an extensive list of conversational systems applied to different domains and languages can be found on chatbots.org [1].

In the next section, we will analyze how NLIDBs are usually classified regarding their architecture, and we will show the advantages and disadvantages of each one of them.

5.3 Technologies and architectures

Natural Language Interfaces to DataBases systems (NLIDBs) are usually classified according to their architecture (Androutsopoulos et al., 1995; Pazos R. et al., 2013). In this way, we can find four types of systems: pattern matching, syntax-based, semantic grammars, and intermediate representation languages. In this section, we will analyze all these architectures, showing their advantages and disadvantages.

Although in this chapter we will only consider this approach regarding the architecture, some authors (Pazos R. et al., 2013) also include additional classifications according to the language processing technique (e.g. symbolic approach —words are symbols representing objects and concepts in the real world—, empirical approach —statistical analysis of documents—, and connectionist approach —based on neural networks—) or the type of graphical user interface (GUI) (e.g. command line, dynamic menus, conversations and multimode).

5.3.1 Pattern matching systems

Pattern matching is the typical architecture used in first NLIDBs. It basically uses two main concepts, patterns and expressions. On the one hand, a pattern is a simple template which can be used to match the user's input in order to

[1] https://www.chatbots.org/

identify the type of question. It can be a little bit more complex if, for example, words are reduced to their roots or synonyms are used. On the other hand, an expression is a query, usually written in SQL language, associated to the kind of question and used to get the desired information from the database.

In order to better understand this approach, we will briefly describe two examples of patterns and expressions that can be found in Androutsopoulos et al. (1995). If we consider the pattern "`... capital ... <Country>`", it could have the associated expression "`SELECT capital FROM country WHERE country = <Country>`". This would retrieve the capital of a specific country (e.g. "*What's the capital of Spain?*"). On the other hand, the pattern "`... capital ... country ...`" could have the associated expression "`SELECT country, capital FROM country`". This second pattern would answer questions like "*List the capital of every country.*".

The main advantage of a pattern matching system is its simplicity. It is really easy to implement, add, and remove features from the system, and there is no need for syntactic parsing. In addition, sometimes this kind of systems can provide a reasonable answer when the user asks a question that is out of the range of sentences the patterns were designed to handle. For example, in the face of the question "*Is Barcelona the capital of Spain?*", the system might not provide a direct "*no*" answer to the question but a list containing the actual capital of Spain, Madrid. This is not the ideal answer, but it is okay.

However, this kind of systems presents a huge disadvantage, its superficiality. When somebody talks with one of these systems, she suddenly realizes that many mistakes are made, and the coverage is limited by the number of patterns. For these reasons, they are not enough for making a good natural language interface to databases system.

Some examples of pattern matching system are gNarLI (Shankar and Yung, 2000).

5.3.2 Syntax based systems

A syntax based system analyzes the sentence to make a syntactic (parse) tree which is translated into predicate logic to get the database query. These systems

`S` → `WQ VP`
`WQ` → `WH` *river* | `WH` *state*
`WH` → *what* | *which*
`VP` → `V DP`
`V` → *borders* | *passes through*
`DP` → *Illinois* | *Indiana* | *Missouri*

(a) Syntactic tree

`S` → `RiverQuestion FlowThrough State`
`RiverQuestion` → (*what* | *which*) *river*
`FlowThrough` → *passes through*
`State` → *Illinois* | *Indiana* | *Missouri*

(b) Semantic tree

Figure 5.2: Trees for question "*Which river passes through Illinois?*"

use grammar rules and lexicons. A lexicon is essentially a catalog containing the words of the language, whereas grammar rules combine those words into meaningful sentences in that language.

Taking the GeoBase [1] database and considering the question "*Which river passes through Illinois?*" adapted from Pazos R. et al. (2013), using a lexicon and some grammar rules, the system could make the syntactic tree shown in Figure 5.2a. This parse tree would be mapped to the logical query `?(river(x) ∩ flow_through(x, Illinois))`.

The main advantage of this kind of systems is that they provide a detailed structure of the sentence (part-of-speech tagging: verb, noun, adjective, preposition, adverb...), and that syntactic tree can be easily mapped to the database query.

Nevertheless, these systems presents some drawbacks such as semantic ambiguity (a column belonging to several tables), several interpretations (syntactic trees) for a unique query, bad portability (rooted to database), and it is not always clear when a node should add semantic information.

The typical example of syntax based system is LUNAR (Woods et al., 1972), although more recent systems such as FREyA (Damljanovic et al., 2012) and NaLIR (Li and Jagadish, 2014b) also make use of use syntactic parsing and

[1] http://www.geobase.ca/

analysis.

5.3.3 Semantic grammar systems

Semantic grammar systems are very similar to syntax based systems. In this case, they simplify the parse tree to the minimum removing or combining the nodes that are not needed for making a semantic interpretation of the query. In this way, they better reflect the semantic representation without having complex tree structures.

Since semantic grammar categories are selected to enforce semantic constraints, a grammar or production rule here might not correspond to general syntactic concepts. For example, if we consider the same question as before, "*Which river passes through Illinois?*", it could be parsed into the semantic tree that appears in Figure 5.2b (Pazos R. et al., 2013). As we can see, this semantic tree is shallower and easier to understand and process than the corresponding syntactic tree of Figure 5.2a.

The main advantage of this kind of systems is its great performance. There is no need for complex syntactic trees. In addition, the semantic information is assigned to tree nodes, so it reduces the elliptical problems of user queries (when a user omits some words of the sentence needed for a correct grammatical reconstruction but not for understanding its meaning).

However, these systems are difficult to port. New semantic grammars are needed for new domains, although some systems try to make these rules automatically from a corpus or interacting with the user.

Some examples of semantic grammar systems are REL (Thompson et al., 1969), PHLIQA1 (Scha, 1977; Bronnenberg et al., 1980), LADDER (Hendrix et al., 1978), PLANES (Waltz, 1978), EUFID (Templeton, 1979; Templeton, 1983), ASK (Thompson and Thompson, 1983), DATALOG (Hafner, 1984), CoBase (Zhang et al., 1999), JUPITER (Zue et al., 2000), MERCURY (Seneff and Polifroni, 2000), NLA (Budzikowska et al., 2001; Chai et al., 2001b; Chai et al., 2001c; Chai et al., 2001a; Chai et al., 2002), PRECISE (Popescu et al., 2003), WASP (Wong and Mooney, 2006), and Let's Go (Raux et al., 2006).

5.3.4 Intermediate representation languages

Together with semantic grammars, this is one of the most used techniques. Instead of directly translating the user's query into SQL using a syntax-based approach, they use an intermediate language of logic queries (predicate logic), which is then translated into a specific database query language.

The main advantages are the independence of the database management system (DBMS), and the possibility of including reasoning modules between the semantic analyzer and the database query generator.

Nevertheless, these systems have a limited application scope, since most of them use deductive databases, much less used than relational databases.

Some examples of intermediate representation languages systems are RENDEZVOUS (Codd, 1974), CHAT-80 (Warren and Pereira, 1982), TELI (Transportable English-Language Interface) (Ballard and Stumberger, 1986), TEAM (Grosz et al., 1987), JANUS (Hinrichs, 1988), MASQUE/SQL (Androutsopoulos et al., 1993), Edite (Reis et al., 1997), ORAKEL (Cimiano et al., 2007b; Cimiano et al., 2007a; Cimiano et al., 2008), and C-Phrase (Minock et al., 2008; Minock, 2010).

5.4 SmartSeller, our proposal

Some authors (Pazos R. et al., 2013) state that a good natural language interface to databases (NLIDB) system should: be easy to configure and use; include tools for modifying the knowledge; make its capabilities and limitations evident to users; offer recommendations sufficiently justified; be robust in case of possible failure; answer quickly and with accuracy; answer deductive, temporal and fuzzy queries; be multimodal; be independent from the domain, the database management system, the language, the hardware, and the software; handle linguistic phenomena (e.g. anaphora, ellipsis, ambiguity, or incomplete search values). Moreover, if we focus on the context of that broader sense of NLI described in Section 5.1 and depicted in Figure 5.1, we think that not only the aforementioned aspects are necessary, but features like interactivity (including a memory and a dialog manager to engage in conversation with users in order to modify or revise

requests), versatility (ability to work with heterogeneous data sources at the same time), emotions (representation using a 3D character able to emotionally react during the interaction), personal advising (by means of recommendations), or business intelligence (integrating rules about business logic) are also important.

As can be seen in Figure 5.3, our system has been designed following a client-server multi-agent architecture. Two types of agents have been defined: expert agents and decision agents. Expert agents are responsible for providing information about specific domains. The system relies on them to be able to answer questions related to those domains. The access to those expert agents is carried out by means of decision agents, which send the user's question to the expert agents, and receive a set of answers from them. In that way, they manage all the answers provided by the different expert agents to provide a unique coherent final answer to the user. Decision agents could be organized hierarchically to make the process of answer integration easier.

This architecture entails many advantages over other NLIDB systems. Its multi-agent nature makes it easily scalable and allows following a divide and conquer approach. Several expert agents are aimed to different purposes (e.g. one agent to resolve general questions and another to resolve product queries). Each agent is an expert doing its own work, and it does not have to know anything about all the other expert agents. They are able to access, process and abstract the information of heterogeneous data sources and architectures to work with reusable high-level semantic interpretations independent from the database management system. These expert agents are connected through decision agents which are able to reason and resolve conflicts in order to get and return the best possible answer to the user. This philosophy allows reducing the overall complexity of the system and improving its performance.

This generic architecture could be applied to any domain. In this chapter, we present a specific design for a bookstore. We include two expert agents and one decision agent, as can be seen in Figure 5.4.

The first expert agent is a Virtual Assistant similar to Elvira [1], the virtual assistant of the University of Granada [2] (Eisman et al., 2012) described in Chap-

[1] http://tueris.ugr.es/elvira/?lang=en
[2] http://www.ugr.es/

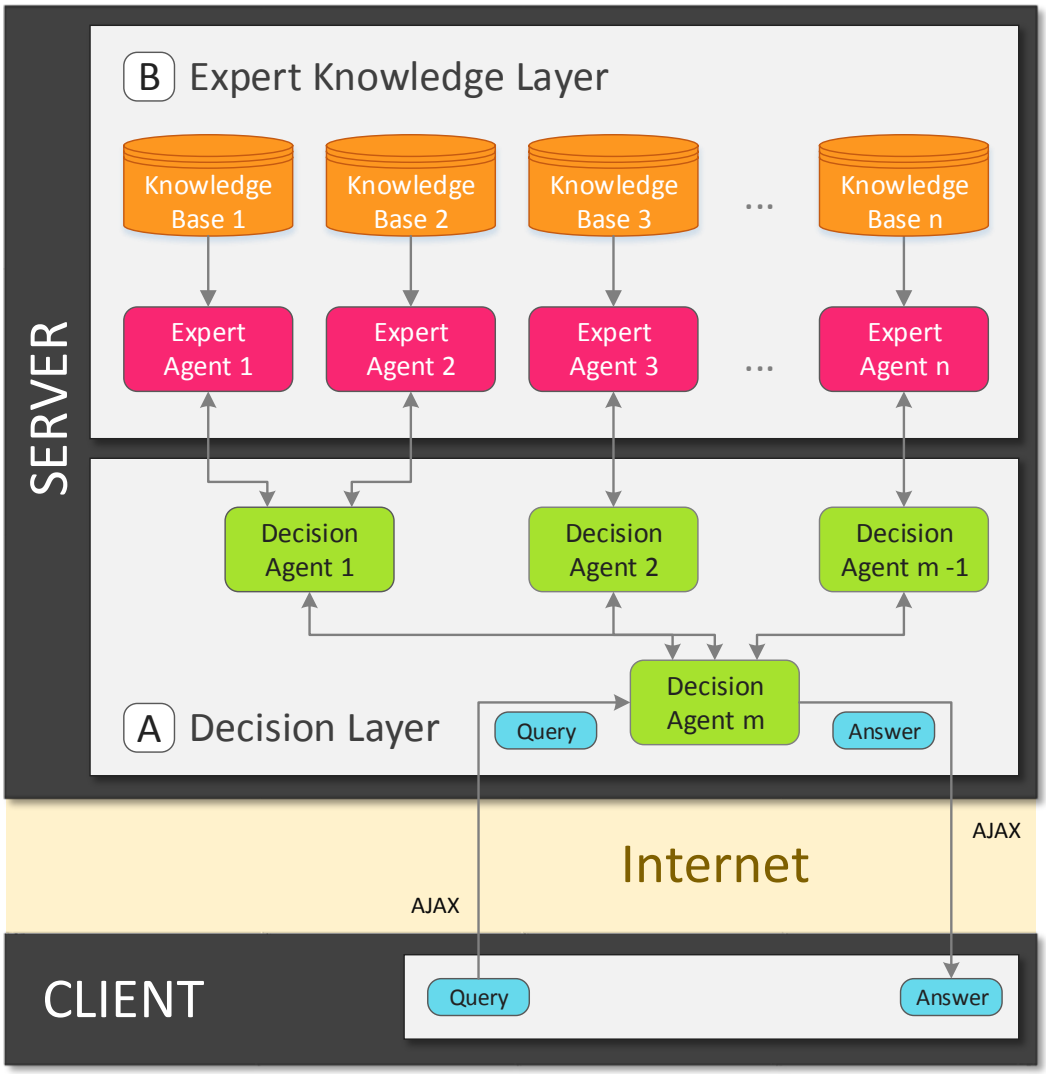


Figure 5.3: Generic Multi-Agent Architecture Overview

ter 4. It is able to answer general questions (e.g. "*What's the time?*", "*What's the weather like?*"), personal questions about the Virtual Assistant (e.g. "*What's your name?*", "*Do you like singing?*"), and specific questions about the shop (e.g. "*What's the opening hours?*", "*What's your telephone number?*", "*How could I pay the book?*"). For the sake of modularity, we have built a hierarchy of ontologies to store the knowledge associated to these three types of questions. In this

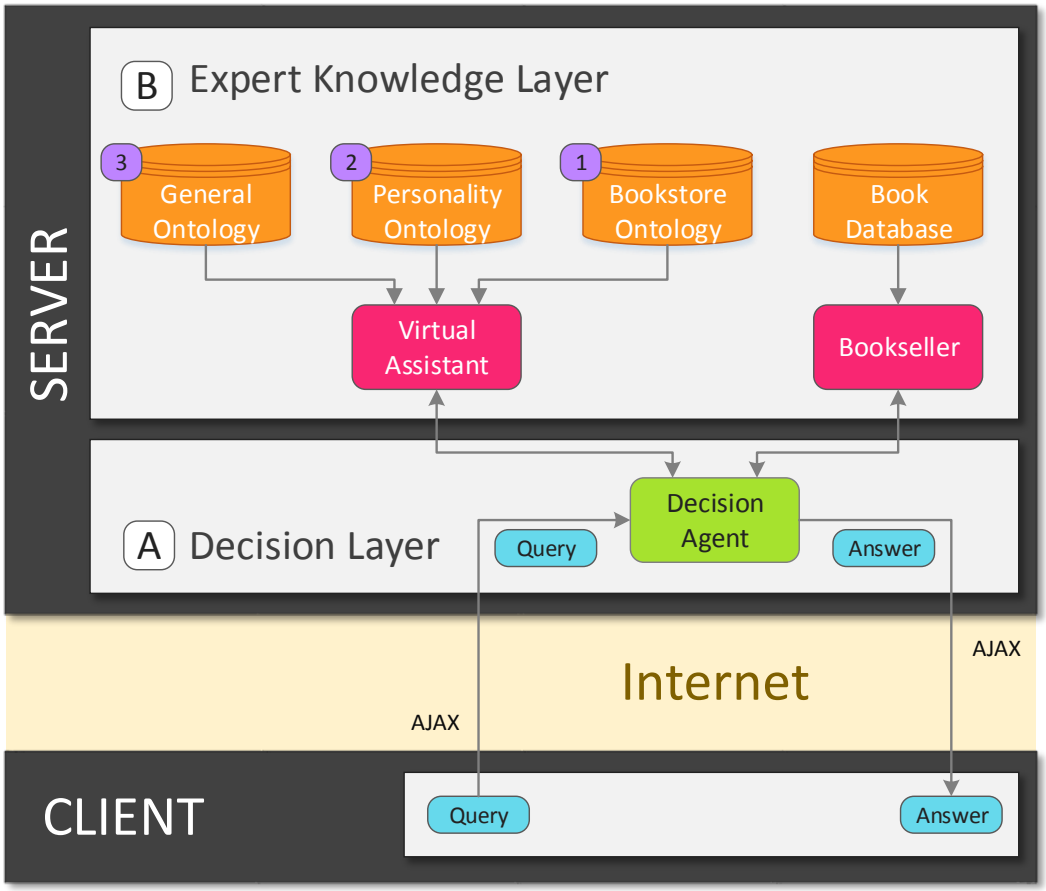


Figure 5.4: SmartSeller Specific Architecture Overview

way, they can be easily connected to and disconnected from the system to solve different tasks or problems. We will talk more about the Virtual Assistant in Section 5.4.1.

The second expert agent is the Bookseller strictly speaking. It provides specific information about a database containing more than two hundred thousand books (titles, authors, publishers, categories, prices, dates...). It uses a semantic grammar and a set of indexes to improve the performance of the system. We will talk more about the Bookseller in Section 5.4.2.

On the other hand, the Decision Agent coordinates both expert agents to provide a unique coherent final answer to the user. We will talk more about the

Decision Agent in Section 5.4.3.

Regarding the languages and technologies, the core of the system has been implemented in Java. The knowledge of the Virtual Assistant is stored in ontologies which are edited using Protégé [1] (Noy et al., 2001) and handled by the system using the Apache Jena [2] library. The emotional state of the avatar is controlled using the fuzzy rule-based system (Eisman et al., 2009a) described in Chapter 3. The Bookseller manages a set of indexes over the book database using Apache Lucene [3]. Finally, the answers provided by the system are transformed into spoken dialog using Nuance Loquendo TTS [4].

5.4.1 The Virtual Assistant

The first expert agent is the Virtual Assistant, which answers general domain questions (e.g. "*What day is it today?*"), personal questions about the Virtual Assistant (e.g. "*How old are you?*"), and general questions about the bookstore (e.g. "*Where is the bookstore?*"). As can be seen in Figure 5.5, it is composed of four main modules. The Natural Language Understander processes the question to generate a list containing all the information units (IUs) to which it refers (this concept will be explained in Section 5.4.1.1). The Dialog Manager uses a rule-based system to filter that list and decide about which IU the Virtual Assistant must talk. The Emotional State Controller updates its emotional state. Finally, the Communication Generator retrieves and adapts the answer for that IU. This answer is made up of several elements: the sentence, the URL of a web page with additional information, a list of recommendations about related topics that can be used to continue the conversation, and so on. Here, we will briefly explain these modules of the Virtual Assistant since they have been already detailed in Chapter 4.

[1] http://protege.stanford.edu/
[2] https://jena.apache.org/
[3] http://lucene.apache.org/
[4] http://www.nuance.com/for-business/customer-service-solutions/loquendo-small-business-bundle/index.htm

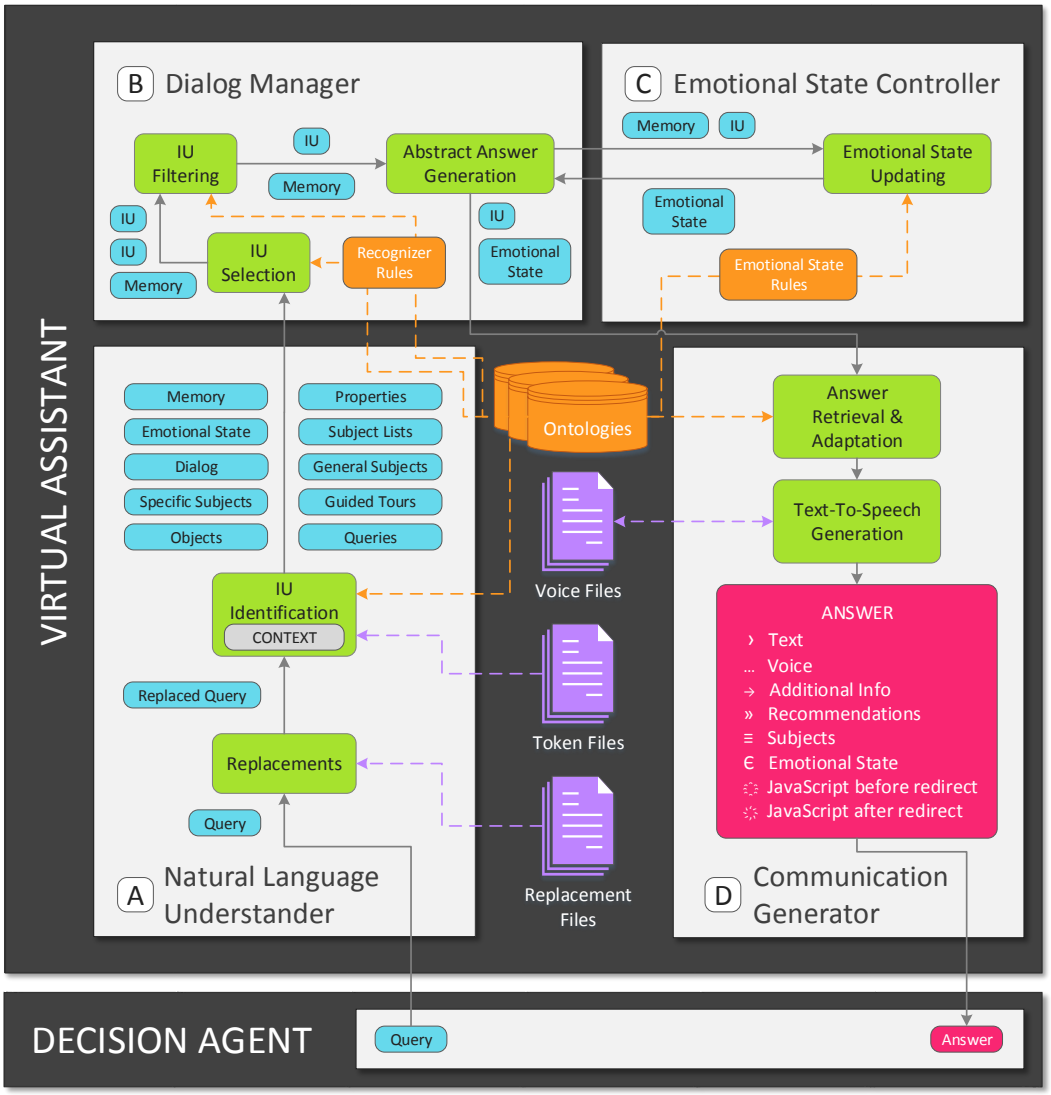


Figure 5.5: Architecture of the Virtual Assistant

5.4.1.1 Knowledge representation

The Virtual Assistant uses two types of knowledge: several plain text files containing thousands of regular expressions to recognize the questions, and a hierarchy of ontologies to store all the information related to the answers provided.

An ontology (Gruber, 1993; Staab and Studer, 2009) is essentially a formal

representation of a set of concepts within a particular domain and some relationship established between those concepts. Our ontology is based on entities that we have called information units (IUs), which are pieces of information about specific concepts. They include the definition of the concept, the name, a URL with additional information, and so on. There are different types of IU, which have been explained in Section 4.3.1: objects, properties, specific subjects, general subjects, subject lists, and guided tours. All these IUs are not isolated in the ontology, but they can be connected with each other. In this way, we can see our ontology as a kind of graph in which the IUs are the nodes and the connections between them are the edges of the graph.

All this information is processed by our own reasoning engine which uses a system of configurable rules, temporal restrictions, context, memory, and many other interesting features explained in Chapter 4.

The other type of knowledge is all the information needed for the recognition of the different questions. This data is stored using plain text files which contain tokens associated to the IUs. These tokens represent patterns or regular expressions that capture all the different ways that people use to refer to them, and the initial set is specified by hand for each language. Figure 5.6 shows some very simple examples. Although they are less precise than other types of recognition techniques like semantic grammars, they entail an easy way to capture knowledge.

This could become a very tedious work, but it can be simplified using wild cards (written as *). This is a greedy operator that can represent any set of words, so the number of possibilities to be written is reduced. However, it must be used carefully to avoid capturing wrong sentences.

Additionally, Figure 5.6 shows how we can specify different ways for referring to a certain property that will only be taken into account in the context of a specific class. For example, "*home line number*" could be used to refer to the telephone number of a person but not for a company.

Another simplifying mechanism is the use of replacement files including regular expressions about certain misspellings that can appear in the question, synonyms, or complex IU names. This reduces the number of words of the question making the processing of the token files faster, increases the reliability of the understanding process since it reduces the ambiguity of questions, and simplifies

$Bookstore:\quad compan(ies|y) \mid firms? \mid books?(\ |-)?(shop|store)s?$

$Address:\quad address(es)? \mid addrs? \mid a?venues? \mid directions? \mid loca(liza)?tions? \mid placements? \mid streets?$
$Address:\quad how\ *\ (access \mid arrive \mid find \mid get\ to \mid locate)$
$Address:\quad (what \mid where \mid which)\ *\ (a?venues? \mid directions? \mid streets?)$

$Company\_Address:\quad \#Property : Address$
$Company\_Address:\quad where\ *\ (offices? \mid (head)?quarters)$

$Person\_Address:\quad \#Property : Address$
$Person\_Address:\quad where\ *\ (live \mid work)$

$Telephone:\quad blower \mid cellulars? \mid (smart)?phone \mid tel(ephone)s?$
$Telephone:\quad (contact \mid information)\ numbers?$
$Telephone:\quad numbers?(\ of)?\ (contact \mid information)$

$Company\_Telephone:\quad \#Property : Telephone$

$Person\_Telephone:\quad \#Property : Telephone$
$Person\_Telephone:\quad home(\ ?line)?\ numbers?$
$Person\_Telephone:\quad android \mid iphone \mid motorola \mid nokia$

Figure 5.6: An excerpt from a token file

the maintenance of the system.

On the other hand, there are some tools that have been designed to reduce the effort devoted to this task. For example, Moreo et al. (2013) proposed a semiautomatic method to reduce the problem of creating templates to that of validate, and possibly modify, a list of proposed templates. In this way, a better trade-off between reliability —the system is still monitored by an expert— and cost is achieved. In addition, updating templates after domain changes becomes easier, human mistakes are reduced, and portability is increased.

5.4.1.2 The Natural Language Understander

The Natural Language Understander identifies all the information units (IUs) referred by the question. First, it reduces the language to a specific vocabulary so that it can be managed more easily. Some misspellings are corrected, some

words are replaced with synonyms, some complex names with abbreviations, and so on. Next, it uses the token files to generate a list with all the IUs referred by the replaced query.

For example, let us consider the question "*What is the address of the bookstore?*". First, some expressions would be replaced giving as a result the query "*address the bookstore*". This replaced query would be used to look for the IUs in the token files. In this case, the object *Bookstore* and the property name *Company_Address* would be identified and passed to the Dialog Manager, which would generate an appropriate answer. In this case, the answer would be the property *Bookstore_Address*.

5.4.1.3 The Dialog Manager

The Dialog Manager uses the list of information units that matches the user's question to make an abstract answer that is later transformed into a specific answer by the Communication Generator (CG). The process consists of two main steps, IU selection and filtering. Selection is a rule-based process that allows changing the provided answer depending on if it contains a certain IU, if the Virtual Assistant is in a certain emotional state, etc. Filtering removes ambiguity using the contextual information that can be extracted from the ontologies.

The remaining IUs (at least one) are used to generate the abstract answer that is used by the CG to create a specific answer, as will be described in Section 5.4.1.5.

5.4.1.4 The Emotional State Controller

The Emotional State Controller manages the emotional state of the Virtual Assistant using the dynamic probabilistic fuzzy rule-based system (Eisman et al., 2009a) described in Chapter 3. As we saw in Section 3.3, it uses two essential concepts: emotional state and personality. Emotional state is built on the basis of eight basic emotions that correspond with the following emotional attributes: joy, disdain, anger, fear, worry, surprise, sadness, and embarrassment. The value of each attribute is a real number between 0 (total absence) and 1 (total presence). On the other hand, personality is a set of static features that allows carrying

```
AntecedentAnswer[Answer == ComplimentGoodLooking]
∧ AntecedentMemory[ComplimentGoodLooking == Zero]
⇒ ConsequentEmotionalState[EmbarrassmentVariation = Low_Positive]
```

Figure 5.7: Example of fuzzy rule to control the emotional state variation

out a subjective assessment of the questions asked by the user, and that makes the emotional state tend to an equilibrium. Six different personalities have been defined: anguished, depressive, hypochondriac, maniac, phobic, and normal.

An example of rule is shown in Figure 5.7. It states that IF the selected IU to answer the question is `ComplimentGoodLooking`, AND this IU is not already present in the memory of the Virtual Assistant (it is the first time the user makes a compliment), THEN a low positive variation of the embarrassment emotional attribute is produced.

5.4.1.5 The Communication Generator

The Communication Generator (CG) instantiates the abstract answer provided by the Dialog Manager in a concrete answer written in a specific language. The core of the answer is a sentence associated to the IU that can change depending on a set of constraints like the time or the date. It is also transformed into spoken dialog using the Nuance Loquendo text-to-speech. In addition, it can include a web page where the user can find additional information, and a recommendation list with some other IUs related to the answer, which might be very interesting for the user.

5.4.2 The Bookseller

The Bookseller is a domain specific expert agent that answers questions about a database containing thousands of books. It differs from the Virtual Assistant in many aspects. In general, although a natural language understanding based on tokens is quite simple and works more or less well in some circumstances (even providing reasonable answers for some out-of-domain questions), it does not help the system to really understand the meaning of the question asked by the user, and this superficiality leads it to make many mistakes. However, taking

advantage of the structure and the semantics of the question, the system can better handle ambiguity, provide more precise answers, and improve the quality of the recognition. For this reason, although in Figure 5.8 we can identify the same module names as the ones of the Virtual Assistant, they are completely different. The Bookseller employs a dynamic semantic grammar which allows making conditions of different classes, and a set of indexes which makes easier the access to the information of the database. In addition, the grammar generates an intermediate representation language that makes the system independent of the database management system (DBMS). Next, we will describe the whole process:

1. The Bookseller gets a new query from the Decision Agent.
2. The Natural Language Understander (NLU) clones the original base grammar, which does not contain any specific rule about titles, authors, or any other database attribute. These specific rules are added afterwards.
3. The NLU filters the production rules of the cloned grammar removing the final tokens that do not match any fragment of the question. This speeds up the recognition process.
4. The NLU uses the question to look for titles, authors, categories, publishers, and descriptions in the indexes, and labels in the concepts hierarchy. For the retrieved values, specific rules are generated and dynamically added to the grammar.
5. The NLU carries out a fast pass over the grammar to filter out all the production rules that are not able to generate any part of the question.
6. The NLU processes the grammar to get all the possible restrictions specified by the question.
7. The NLU filters the restriction list using a set of predefined files in order to remove noisy restrictions.
8. The Dialog Manager (DM) processes the restriction list to generate all possible valid interpretations to the question.

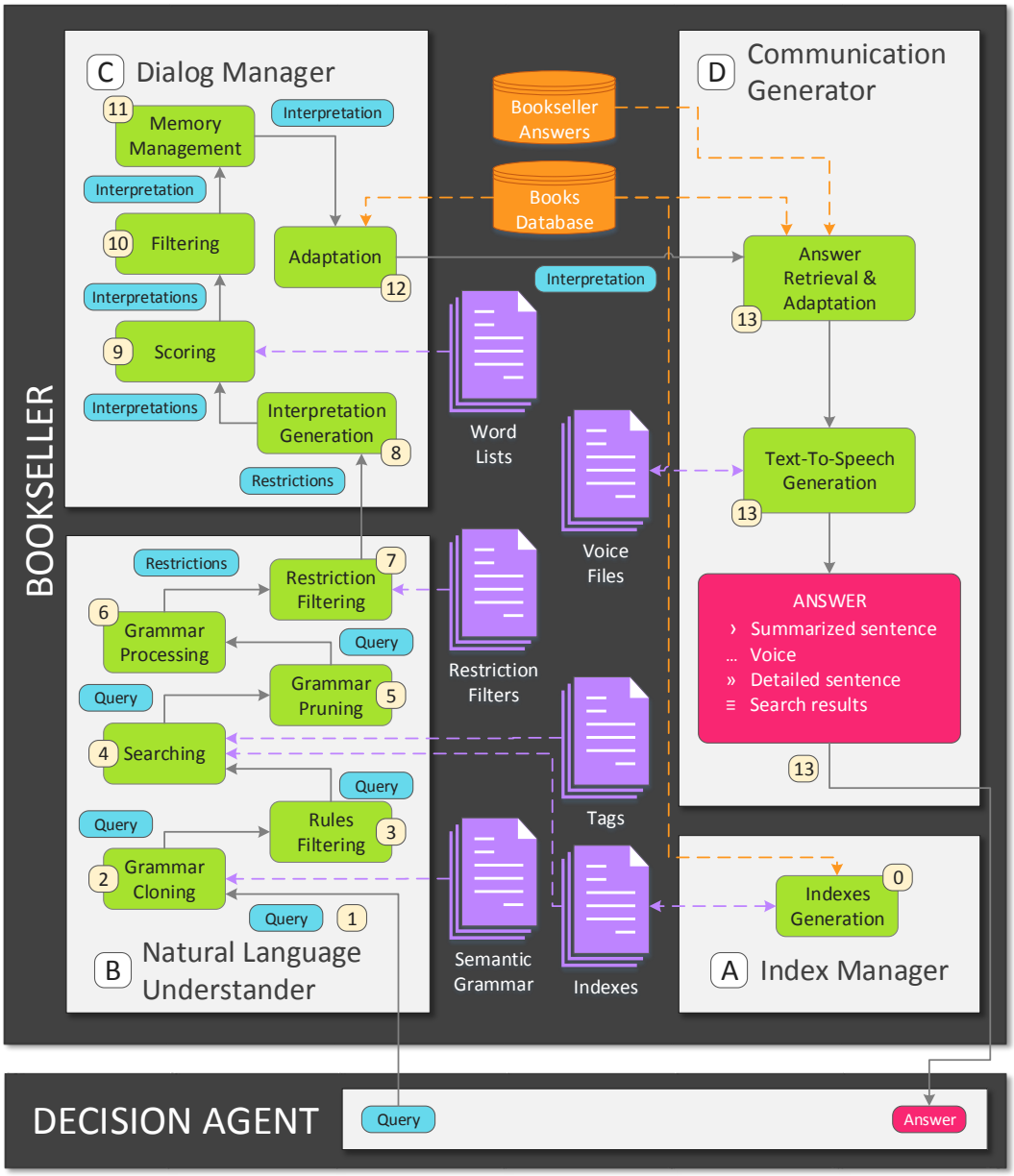


Figure 5.8: Architecture of the Bookseller

9. The DM classifies these interpretations using scores.

10. The DM bets for the best interpretation considering the type of restrictions.

11. The DM determines if compatible restrictions from the memory can be reused.

12. The DM adapts the chosen interpretation to always retrieve at least one record.

13. The Communication Generator (CG) makes a specific answer for the interpretation selected by the Dialog Manager and passes it to the Decision Agent.

Next, we will analyze all these details of the Bookseller.

5.4.2.1 Knowledge representation

All the information that the Bookseller is going to provide is stored in a database of a real bookstore containing more than two hundred thousand books. Title, author, publisher, category, date, and price are just a few examples of all the information that we can get for each book. A set of indexes makes the access to the database easier, and different operators define how these indexes are created. In addition, a labels hierarchy and a dynamic semantic grammar recognize and capture all the valuable information to provide a precise answer.

The Indexes In order to facilitate the portability of the system, we access the database through a view that contains all the needed fields. Once defined, the Natural Language Understander module accesses it using a set of indexes handled by the Index Manager. In this way, the accuracy and performance of the system are increased since we can define operators for preprocessing the information obtained through the view. This allows making partial or incomplete searches (e.g. using only few words of the title, or the initials of the author), and it resolves some consistency problems that data might have.

The Dynamic Semantic Grammar It allows creating a structure of restrictions and interpretations that are ultimately translated into the particular query language of the database. It is modified at runtime to remove the unnecessary

grammar rules for a given query and add specific rules using the indexes and labels.

The initial set of grammar rules is specified by hand for each language, and it is fixed and improved with the use of the system. At this moment, our grammar consists of approximately one hundred and fifty semantic rules organized in five different categories: restrictive rules that impose restrictions over specific attributes of the database, sorting rules that order the retrieved records by a specific attribute, top rules that get top or bottom records once they have been sorted according to some criterion, delete rules that remove previously imposed restrictions, and void rules that capture common text expressions that improve the recognition process. Next, we will show some simplified examples of each category.

Restrictive rules They allow searching by `TITLE` (exact or partial matching), `AUTHOR` (exact, partial, or using initials), `PUBLISHER`, `CATEGORY`, `LABEL`, publication `DATE` (exact year, interval, fuzzy categories such as recent, very recent, or not very recent), `PRICE` (exact price, interval, cheap, expensive), `ISBN`, and `EAN`.

For example, if we asked the question "*books written by Shakespeare*", and considering the start symbol ∼`CONDITION`, the three grammar rules of Figure 5.9 would be able to produce the fragment "*written by shakespeare*" and impose the restriction `AUTHOR = "shakespeare"`. The first rule states that a ∼`CONDITION` could be formed by an optional ∼`BY_AUTHOR` (note the question mark after the symbol) followed by a mandatory ∼`AUTHOR_VALUE`. The second rule states that ∼`BY_AUTHOR` represents any sentence with the same meaning as "*written by*". Finally, ∼`AUTHOR_VALUE` produces any of the authors contained in the database. Although for the sake of clarity we have only included the author Shakespeare in this third rule, in Section 5.4.2.2 we will see how this type of rules are dynamically generated depending on the concrete question made by the user.

Another interesting feature is the possibility of associating statements to any production rule. Figure 5.9 shows three types of statements. The first one is associated to the ∼`AUTHOR_VALUE` rule and it has the form ∼`author = #value#;`. It allows defining variables to capture knowledge from users. In this case, it states that the complete value captured or produced by this rule

```
~CONDITION → ~BY_AUTHOR? ~AUTHOR_VALUE;
    @smartseller.condition.String(AUTHOR, EQUALS, ~author);

~BY_AUTHOR → (by|of)( the)?( (author|poet|writer)s?)?  ||
              somebody (named|whose name is)  ||
              ((that|which) )?((ha(s|ve)( been)?|w(as|ere)) )?
                  (ma(de|ke)|writ(e|o|ten))( by)?( the)?
                  ( (author|poet|writer)s?)?  ||
              (the |which )?(author|poet|writer)s?( (are|is|was|were))?;
    @%CAPTURE_TEXT = false;

~AUTHOR_VALUE → shakespeare;
    ~author = #value#;
```

Figure 5.9: Restrictive rule for `AUTHOR:"shakespeare"`

is stored in a variable named `author`, which is used to generate conditions later on. The second statement is associated to the ~`BY_AUTHOR` rule and it is `@%CAPTURE_TEXT = false;`. This means that the text produced by this rule is irrelevant for the condition that is generated. In contrast, the text produced by ~`AUTHOR_VALUE` is meaningful since it is the name of the author and it should be shown to the user to justify why certain restrictions have been imposed over the data. Finally, the last statement is associated to the ~`CONDITION` rule and it is `@smartseller.condition.String(AUTHOR, EQUALS, ~author)`. It means that, if this rule produces some non-empty fragment of the question, then it creates a `String` condition over the `AUTHOR` attribute using the value captured by the `author` variable in the ~`AUTHOR_VALUE` rule. Although this only shows the use of a `String` condition, many other data types have been implemented and can be used: `Array`, `Date`, `Double`, `Integer`, and so on.

It is important to emphasize that production rules for the start symbol of the grammar ~`CONDITION` do not need to produce the whole question but only fragments of it. This is essential to make a robust system which should do its best from the question although it be not grammatically perfect. Nevertheless, the more text is produced by the grammar, the better for the recognition process since less ambiguity is generated and more accuracy is gained.

Figure 5.10 includes another interesting example of grammar rule which shows

```
~CONDITION → ((in|of) )?(th(e|is) )?(current|present) year  ||
             ((in|of) )?this year;
    @smartseller.condition.Date(false, DATE, GREATER_OR_EQUALS,
        #time_this_year_first_day#, LESS_OR_EQUALS,
        #time_this_year_last_day#, null);
```

Figure 5.10: Restrictive rule for `DATE:"this year"`

```
~CONDITION → ~LAST ~NUMBER ~MONTH;
    @smartseller.condition.Date(false, DATE, GREATER_OR_EQUALS,
        -~number, null, null, MONTH);

~LAST → last;

~NUMBER → \d+;
    @~number = java.lang.Integer(#value#);

~MONTH → months?;
```

Figure 5.11: Restrictive rule for `DATE:"last months"`

the use of temporal questions. They allow users to refer to relative dates, so the generated restrictions depend on the time when questions are asked. In this way, the same question could generate different restrictions in different moments. For example, the question "*books published in this year*" would impose a `DATE` restriction with an interval between the first day and the last day of the current year. This values, specified in the rule by `#time_this_year_first_day#` and `#time_this_year_last_day#`, are instantiated at runtime.

Another example of rule covering temporal questions is the one shown in Figure 5.11. We can see how a `~NUMBER` rule stores a number of months in a variable with the same name. This value is later used to create a `Date` condition for the last `~number` months (note the minus sign before this variable in the `~CONDITION` rule). In the same way, Figures 5.12 and 5.13 show how additional rules can cover date intervals or common time expressions such as "*last decade*".

As we can see, these kind of rules are reusable and can be effortlessly ported to other application domains.

Other restrictive rules include parameters instantiated at runtime that can

```
~CONDITION → ~BETWEEN ~DATE_VALUE_BOTTOM ~AND ~DATE_VALUE_TOP  ||
             ~DATE_VALUE_BOTTOM ~OR ~DATE_VALUE_TOP;
    @smartseller.condition.Date(false, DATE, GREATER_OR_EQUALS,
        ~bottom_date, LESS, ~top_date, null);

~DATE_VALUE_BOTTOM → ~DATE_VALUE;
    @~bottom_date = java.lang.Calendar(#time#);

~DATE_VALUE_TOP → ~DATE_VALUE;
    @~top_date = java.lang.Calendar(#time#);

~DATE_VALUE → ~DAY_VALUE ~DATE_SEPARATOR ~MONTH_VALUE
                  ~DATE_SEPARATOR ~YEAR_VALUE;

~DATE_SEPARATOR →   .  ||  -  ||  \  ||  /  ||  of;

~DATE_VALUE → ~DETERMINED_ARTICLE? ~YEAR? ~YEAR_VALUE;

~YEAR → years?;

~YEAR_VALUE → \d{2};
    @~year = java.lang.Integer(19#value#);

~YEAR_VALUE → 19\d{2}|20\d{2};
    @~year = java.lang.Integer(#value#);

~MONTH_VALUE → 0?[1-9]  ||  1[0-2];
    @~month = java.lang.Integer(#value#);

~MONTH_VALUE → jan(uary)?;
    @~month = java.lang.Integer(1);

~DAY_VALUE → 0?[1-9]  ||  [1-2]\d  ||  3(0|1);
    @~day = java.lang.Integer(#value#);
```

Figure 5.12: Restrictive rule for `DATE: (date interval)`

be customized to modify the behavior of the system (when a book is considered recent, cheap, and so on). For example, the owner of the bookstore could consider that a book is cheap if it costs ten euros or less, whereas it is very cheap if it costs five euros at the most. On the other hand, a book could be considered recent if

```
~CONDITION → last decade;
    @smartseller.condition.Date(false, DATE, GREATER_OR_EQUALS,
        -10, null, null, YEAR);
```

Figure 5.13: Restrictive rule for `DATE:"last decade"`

```
~CONDITION → ((that|which) )?((is|are) )?(brand|clearly|
                 extremely|frankly|really|super|very)
                 (current|modern|new|recent(ly)?)  ||
             late(st)?  releases?;
    @smartseller.condition.Date(false, DATE, GREATER_OR_EQUALS,
        @grammar.date-very-recent, null, null, DAY);
```

Figure 5.14: Restrictive rule for `DATE:"very recent"`

it has been published within last year, whereas it could be very recent if it has been published within last six months.

One of this type of rules is shown in Figure 5.14. It creates a `Date` condition which includes a parameter named `@grammar.date-very-recent`. This parameter specifies when a book is considered very recent and its concrete value is retrieved from a settings table at runtime. In this way, the question "*brand new computers books*" could retrieve the computers books published within last six months, for example.

Sorting rules They retrieve records sorted by specific fields such as date or price. The rules shown in Figure 5.15 state that records are sorted by price in ascending order by default. For example, the question "*poetry books sorted by price*" would impose the restrictions `CATEGORY = "poetry"` and `ORDER BY PRICE ASC`.

In addition to these default sorting rules, others explicitly sort in ascending or descending order.

Top rules They retrieve top or bottom records once they have been sorted by a certain criterion. For example, for the question "*the three most recent books written by Shakespeare*", the rules shown in Figure 5.16 would impose the restrictions

```
~CONDITION → ~PRICE ~SORTED_BY  ||
             ~SORTED_BY ~PRICE;
    @smartseller.condition.OrderBy(PRICE, ASC);

~SORTED_BY → ~BY  ||
             ~SORTED ~BY?;

~BY → according to( (its?|the(irs?)?))?  ||
      by( (its?|the(irs?)?))?;

~SORTED → order(ed|ing|s)?  ||
          sort(ed|ing|s)?;

~PRICE → amounts?  ||
         prices?  ||
         values?;
```

Figure 5.15: Sorting rule for `PRICE`

`AUTHOR = "shakespeare"`, `ORDER BY DATE DESC`, and `LIMIT 3`.

Similar rules allow the system to answer questions like "*the cheapest version of Romeo and Juliet*" or "*the first book by Shakespeare*".

Delete rules They remove previously imposed restrictions. For example, using the rule shown in Figure 5.17, the question "*any other author*" would delete any restriction imposed by some restrictive rule over the `AUTHOR` attribute.

Void rules They capture specific text fragments (e.g. common expressions, requests for something...) that might generate undesirable ambiguity. The objective is to prevent them from being a detriment to interpretations during the calculation of their scores. Figure 5.18 shows some examples of this type of rules.

Labels They are features that we associate to a database record without modifying it. They can be defined manually or automatically using a thesaurus. This enriches the data and fills the knowledge gaps that might exist in the database. In this way, labels are a great value since they allow users to make queries containing information that is not even present in the database.

```
~CONDITION → ~DETERMINED_ARTICLE? ~MOST_RECENT ~NUMBER ~BOOK?
                 ~PUBLISHED?  ||
             ~DETERMINED_ARTICLE? ~NUMBER ~BOOK? ~MOST_RECENT
                 ~PUBLISHED?  ||
             ~DETERMINED_ARTICLE? ~NUMBER ~MOST_RECENT ~BOOK?
                 ~PUBLISHED?;
    @smartseller.condition.Limit(~number);
    @smartseller.condition.OrderBy(DATE,  DESC);

~DETERMINED_ARTICLE → the;

~MOST_RECENT → la(st|test)  ||
               least (ancient|old)  ||
               most (current|modern|new|recent)(ly)?  ||
               newest  ||
               posthumous;

~NUMBER → \d+;
    @~number = java.lang.Integer(#value#);

~NUMBER → three;
    @~number = java.lang.Integer(3);

~BOOK → books?  ||
        comics?  ||
        editions?  ||
        novels?  ||
        releases?  ||
        stor(ies|y)  ||
        tales?  ||
        texts?  ||
        versions?  ||
        works?;
    @%CAPTURE_TEXT = false;

~PUBLISHED → ((that|which) )?(w(as|ere) )?(edited|published|written);
```

Figure 5.16: Top rule for `DATE`

Labels are defined using a hierarchy of concepts. This allows the Bookseller to recommend related books and give advice when a user is not sure about what

```
~CONDITION → ~ANY_OTHER ~AUTHOR;
    @smartseller.condition.Delete(AUTHOR);

~ANY_OTHER → ((by|of|with) )?any( others?)?  ||
             others?;

~AUTHOR → (author|poet|writer)s?;
```

Figure 5.17: Delete rule for AUTHOR

```
~CONDITION → ~COMMON_EXPRESSION;
    @smartseller.condition.Void();

~COMMON_EXPRESSION → by the way  ||
                     good (afternoon|morning|night)  ||
                     hello  ||
                     my name( is)?  ||
                     please  ||
                     so much  ||
                     thank you  ||
                     thanks  ||
                     very much  ||
                     would you mind;
    @%CAPTURE_TEXT = false;
```

Figure 5.18: Void rule for common expressions

product to buy.

A perfect example to illustrate the use of labels is the query "*superheroes*". As can be seen in Table 5.1, each label is determined by three parameters: in the presence of which questions we want the label to fire, which concepts we want to associate to the label, and a mask indicating where the Bookseller should look for those concepts (e.g. 1 for searching through the title, 2 for the description, 3 for both the title and the description, 4 to associate a concrete book using its identifier...).

So, even though the database does not contain any information about when some character is a superhero, the Bookseller knows that Batman and Spider-man

QUESTION	`super ?heroe?s?`
CONCEPTS	`avengers batman captain america hulk iron man marvel spider man spiderman superman superwoman thor wolverine x men`
WHERE	`3 (TITLE & DESCRIPTION)`

Table 5.1: Definition of the "*superheroes*" label

are superheroes, and any book including these words in its title or description would be a good candidate to be retrieved if the user asked "*superheroes*".

5.4.2.2 The Natural Language Understander

The Natural Language Understander identifies the restrictions that are specified by the question. The whole process consists of six main steps: grammar cloning, rule filtering, searching, grammar pruning, processing, and restriction filtering.

Cloning The original grammar is copied to be modified.

Rule filtering Final tokens not matching the query are deleted. For example, all the right parts of the ∼`ANY_OTHER` and ∼`AUTHOR` rules shown in Figure 5.17 would be deleted for the question "*To kill a mockingbird*" since this would not match the words *any*, *other*, *author*, *poet*, and *writer*.

Searching The question is searched using the indexes and labels, and each result is preprocessed to get different combinations which generate new grammar rules. This preprocessing is different for each attribute. For example, Figure 5.19 shows the rules that would be made for the author William Shakespeare permuting his names and initials. Although there could be many combinations, in practice only those appearing in the question are added. What is important is to be flexible enough.

For the title attribute, the preprocessing is a little bit more complex. First, it tries generating combinations for an (almost) exact matching relaxing stopwords

```
~AUTHOR_VALUE → william shakespeare;

~AUTHOR_VALUE → shakespeare william;

~AUTHOR_VALUE → william s;

~AUTHOR_VALUE → s william;

~AUTHOR_VALUE → w shakespeare;

~AUTHOR_VALUE → shakespeare w;

~AUTHOR_VALUE → william;

~AUTHOR_VALUE → shakespeare;
```

Figure 5.19: Dynamic grammar rules for `AUTHOR:"William Shakespeare"`

(using a predefined list) and even omitting probably optional content. For example, it would remove the content in brackets for a book titled "*Around the World in 80 Days (Illustrated Edition)*". If this matching is not successful, it tries to the other side, that is, instead of searching the title returned by the search result through the question asked by the user, it tries an exact matching of the question over the title. This is useful to resolve questions like "*Snow White*" or "*The Seven Dwarfs*", which would be an exact matching over a book titled "*Snow White and the Seven Dwarfs*". Again, stopwords are relaxed for better results. Even so, if any kind of exact matching is not successful, partial matching is tried instead. In a first step, it starts omitting few words in the title. In a second step, it tries matching low frequency isolated words. Figure 5.20 shows some examples of rules that would be generated for the title "*Around the World in 80 Days (Illustrated Edition)*".

Again, only those rules that produce some fragment of the question are generated. In addition, we could use a stemmer to reduce words to their roots or a thesaurus to include synonyms. This would increase the overall performance and accuracy.

```
~TITLE_VALUE → around ~STOPWORD world ~STOPWORD 80 days illustrated
                    edition;

~TITLE_VALUE → around ~STOPWORD world ~STOPWORD 80 days;

~TITLE_VALUE → around ~STOPWORD world;

~TITLE_VALUE → world ~STOPWORD 80 days;

~TITLE_VALUE → 80 days;

~STOPWORD → about || above || after || again || against || all ||
            and || any || as || at || ...;
```

Figure 5.20: Dynamic grammar rules for `TITLE:"Around the World in 80 Days (Illustrated Edition)"`

Pruning Once specific rules have been added to the grammar dynamically, this is pruned to remove those production rules which we know for sure that are not going to match any fragment of the question. This speeds up the grammar processing. For example, if no `~AUTHOR_VALUE` rules were generated, the `~CONDITION` rule shown in Figure 5.9 would be pruned.

Processing All the remaining production rules with a left part equals the start symbol of the grammar `~CONDITION` are processed to extract all possible restrictions from the question.

Restriction filtering Any noisy restriction that might generate undesirable ambiguity is removed. For example, if a title restriction were formed by only one very common word (e.g. "*it*"). Nevertheless, it would be kept if it were formed by more words and the meaning were perfectly clear (e.g. "*the book It*").

5.4.2.3 The Dialog Manager

The Dialog Manager combines restrictions and generates candidate interpretations that are evaluated to select the best answer. The process consists of five

steps: candidate generation, scoring, filtering, memory management, and adaptation.

In this section, we will consider the question "*I am looking for a version of Mr. and Mrs. Smith that costs less than ten euros*", and we will see how it is handled throughout the whole process. We will begin with the assumption that the Natural Language Understander has identified several restrictions for the question: `VOID = "I am looking for a version of"`, `TITLE = "Mr. and Mrs. Smith"`, `AUTHOR = "Smith"`, and `PRICE < 10` euros (justified by "*costs less than ten euros*"), being `VOID` a special variable that captures specific text fragments (e.g. common expressions, requests for something...) that might generate undesirable ambiguity.

Candidate generation Restrictions are combined to generate all possible maximal interpretations to the question. At this step, the Dialog Manager does not care about whether the generated interpretations retrieve any result from the database, it only makes sure that the restrictions are compatible when they make up a new interpretation. Two restrictions are incompatible if they are produced by two fragments of the question that overlap with each other, or if the fulfillment of one implies the non-fulfillment of the other, and vice versa.

Considering the aforementioned question, since the `TITLE` and `AUTHOR` restrictions are not compatible with each other (that author did not write that book), just two maximal interpretations would be generated: `VOID & TITLE & PRICE`, and `VOID & AUTHOR & PRICE`.

Scoring A score is calculated for each maximal interpretation by adding the weights of the words of the question that it cannot produce. The higher this score is, the more penalized the interpretation will be. Three word lists with different weights are used: a generic stopword list (0.2), a domain word list (0.5), and a high frequency word list (0.5). Any other word falling out of these lists has a penalization of 1.0.

In our example, the score of the `VOID & TITLE & PRICE` interpretation would be equal to 0.2, the weight of "*that*", and the one of `VOID & AUTHOR & PRICE` would be 1.4, the sum of the weights of the words "*Mr. and Mrs.*" (0.5, 0.2 and

0.5) and "*that*".

Filtering To choose the best interpretation, these are subsequently sorted by four factors: ascending score, whether it retrieves some results from the database (binary classification), ascending number of restrictions composing it, and type of restrictions (this allows giving more priority to categories over titles, for example). Once sorted, the first complete or sufficient interpretation is chosen. An interpretation is said to be complete if more or less it recognizes the whole question, that is, its score is below a maximum threshold (e.g. 1.0). On the other hand, it is sufficient if, even though it does not recognize the whole question, its restrictions allow considering it good enough. For example, if it includes an `AUTHOR` or `TITLE` restriction and it has a minimum length (e.g. 50% of the question). If no interpretation fulfills these requirements, the question is not recognized by the Dialog Manager.

Following these rules, the best interpretation would be the one containing the `TITLE` restriction. In addition, it would be complete since just one word of the whole question (the "*that*" stopword) would not be produced by that interpretation.

Although at this moment we have just focused on the best interpretation found, the system could be improved if for example the top 3 interpretations were combined when the best one has a pretty high score (i.e. matches only loosely the sentence). However, to prevent users from being confused, the results provided by those interpretations should not be merged but shown independently, as Amazon or Google do when a query does not return any results.

Memory management It is determined whether the selected interpretation is a new query, or whether it refines the previous one and compatible restrictions from memory could be reused. Using a rule-based system, an interpretation is considered a new query if it fulfills one of the following conditions:

- it contains a `TITLE`, `ISBN`, or `EAN` restriction.
- it contains an `AUTHOR` restriction and the previous interpretation already contained an `AUTHOR`, `TITLE`, `ISBN`, or `EAN` restriction.

- it contains a `LABEL` restriction and the previous interpretation already contained a `LABEL`, `CATEGORY`, `TITLE`, `ISBN`, or `EAN` restriction.
- it contains a `CATEGORY` restriction and the previous interpretation contained a `TITLE`, `ISBN`, or `EAN` restriction.
- it contains a `CATEGORY` restriction and the previous interpretation contained a `CATEGORY` and `AUTHOR` restriction.

On the other hand, if the new interpretation is a refinement of the previous one, the Dialog Manager tries to reuse all previous restrictions that are not incompatible with any created by the new interpretation. In this way, the user could make a general query and successively refine it if too many results are returned.

Regarding our example, the chosen interpretation would be considered a new query since it contains a `TITLE` restriction.

Adaptation If the interpretation does not retrieve any record from the database, the Dialog Manager starts to relax restrictions. We prefer to return similar results rather than just answer that no items matched the query. Anyway, the user would be informed about which restrictions could not be satisfied.

Having arrived at this point, two things might happen in our example. If the interpretation returned some results, it would be kept as it was. However, if all the versions of the searched book were more expensive, the `PRICE < 10` restriction would be transformed into an `ORDER BY PRICE ASC` restriction.

5.4.2.4 The Communication Generator

The Communication Generator translates the interpretation selected by the Dialog Manager into a concrete answer that the user can understand. This is made up of a summarized sentence, a sound file, a detailed sentence, and some search results.

Summarized sentence Sentence for the avatar to read that briefly describes the restrictions identified in the last question, and if any of them could not be fulfilled and had to be relaxed.

For instance, if a user asked the question "*The Divine Comedy by Dante*", the Bookseller would make the interpretation `TITLE & AUTHOR`, being the summarized sentence "*Here you can see the books with the title and author you provide me.*". Next, if many results were returned, the user could refine her search with a second question "*a recent version published by Penguin*". The system would combine the new `DATE` and `PUBLISHER` restrictions with the ones from memory, making the interpretation `TITLE & AUTHOR & DATE & PUBLISHER`. If none of the books satisfied the `DATE` restriction, the Bookseller would adapt the interpretation to `TITLE & AUTHOR & PUBLISHER & ORDER BY DATE DESC`, being the summarized sentence in this case "*I haven't found any book with that date restriction, but I show you the ones having that publisher.*".

Voice A phrase dynamically generated by the text-to-speech Nuance Loquendo from the summarized sentence.

Detailed sentence It shows exactly all the restrictions that are being applied, either because they have been recognized from the last question, or because they have been reused from memory.

If we consider the same two questions as before, in the first case the detailed sentence would be "*These are the 21 results that I have found with the title 'The Divine Comedy' and the author 'Dante'.*". On the other hand, the second sentence would contain all the restrictions being applied at that moment, "*These are the 3 results that I have found with the title 'The Divine Comedy', the author 'Dante' and the publisher 'Penguin'.*"

Search results Finally, the answer contains the set of books matching the interpretation of the question and which might be sorted by a certain criterion.

5.4.3 The Decision Agent

The Decision Agent is the joint point between the user and the expert agents. Although the possibility of combining the information provided by different expert agents to make a more complete final answer is very attractive, choosing the response of one expert agent is enough most of the times. For this reason, in this

first version of SmartSeller we have implemented the approach that is described next:

1. The user makes a query which is received by the Decision Agent. This agent sends the question to the Virtual Assistant to see if this expert agent is able to recognize it.

2. If the Virtual Assistant recognizes the whole question, the Decision Agent does not send it to the Bookseller because it already knows the right answer.

3. If the Virtual Assistant does not recognize the whole question, the Decision Agent sends it to the Bookseller.

4. If the Decision Agent just receives a response from one expert agent, that is chosen to answer the question.

5. If both expert agents provide an answer, these are compared by means of a scoring measure like the one used by interpretations, and the best one is chosen to answer the question.

6. The Decision Agent uses an advising module that may change the chosen answer before sending it to the user.

The fact of not sending a question to the Bookseller when the Virtual Assistant is able to perfectly recognize it increases the system performance. However, this optimization affects the way some ambiguous situations are handled because it gives a higher priority to the Virtual Assistant expert agent. For example, the query "*Good morning*" would be recognized as greeting by the Virtual Assistant, so the Decision Agent would not send it to the Bookseller, even though there could be a book with that exact title. In fact, that ambiguity actually happens in Spanish with a book written by Alicia G. García. However, in a dialog system it seems reasonable to prioritize the conversation flow over the search of a very specific book. Anyway, that book could still be found with a more concrete query (e.g. "*I'm looking for the book Good morning*").

5.4.3.1 The Advisor

Sometimes the user may not be sure about what to buy and needs some advice from a shop assistant, and sometimes she may find understanding what kind of questions the system is able to answer to be difficult. For these reasons, we consider that having an advising module to lead the conversation is essential to help people to use the system and, what is more important, get to find what they are looking for.

The Advisor is included in the Decision Agent because it needs to have a general overview of all the expert agents that make up the system. Its main objective is to coordinate them and make the best decision according to the current state of the system and the answer of each agent to the new user's question. This module is based on a finite state machine (Gill et al., 1962; Wagner et al., 2006), a concept that will be explained in the following section.

5.4.3.2 Finite State Machines

Many applications are plain and linear in the sense that they are controlled by a combinational system which produces a set of actions or outputs that only depend on the inputs received at each specific moment. However, this simple approach is not usually enough for modeling complex systems like natural language interfaces. This kind of systems are sequential, that is, the output may depend not only on the present inputs but also on the current state of the system (its past history or sequence of inputs). This behavior can be modeled using a finite state machine (FSM) where all states represent all imaginable situations in which the system may ever be. These states and the transitions between them are usually represented by a transition matrix or a state transition diagram, which will be described next. The most convenient representation format depends on the application being considered.

As can be seen in Table 5.2, a transition matrix is a $\mathtt{n} \times \mathtt{n}$ table, being $\mathtt{n}$ the number of states, where rows represent the state of origin (from) and columns represent the state of destiny (to). In this way, each cell contains the conditions that must be fulfilled to make the transition from the state indicated by its row to the state indicated by its column.

From \ To	Start	State A	State B	State C	End
Start	–	Input 1	Input 2	Input 3	–
State A	–	–	Input 1	Input 2	Input 3
State B	–	Input 2	–	Input 1	Input 3
State C	–	Input 1	Input 2	–	Input 3
End	–	–	–	–	–

Table 5.2: Example of transition matrix with columns representing states

State \ Input	Input 1	Input 2	Input 3
Start	State A	State B	State C
State A	State B	State C	End
State B	State C	Start A	End
State C	Start A	Start B	End
End	–	–	–

Table 5.3: Example of transition matrix with columns representing inputs

An alternative way of representation for transition matrices is the one shown in Table 5.3. In this case, columns represent inputs and rows represent states (or vice versa). Therefore, each cell contains the next state of a transition.

On the other hand, a state transition diagram is a graphical representation equivalent to the aforementioned transition matrix concept. As we can see in Figure 5.21, it is a kind of directed graph where the vertexes (or nodes) are the states of the machine, and the edges (or arcs) are the transitions between those states. Whereas transition matrices are easier to draw, state transition diagrams are easier to understand. Unfortunately, none of these representation models have been designed to handle the actions of a finite state machine, so another representation model is needed.

Before describing this alternative representation model, we must present the concept of action. An action is a task to be performed by the control system when a condition is fulfilled or when a certain input is received. Different types of actions can be defined depending on the conditions and the moment when they

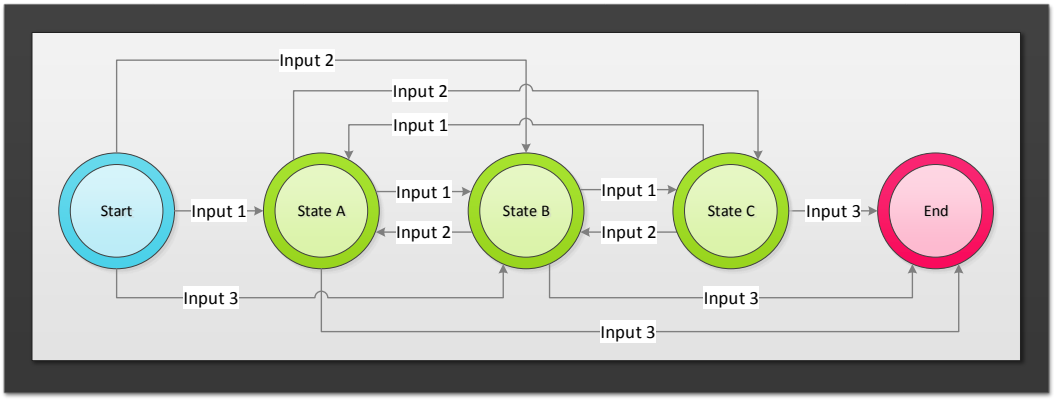


Figure 5.21: Example of state transition diagram

are carried out:

- Input actions: performed when an input condition is true.
- Exit actions: performed when the FSM leaves a state.
- Transition actions: performed during a state change.
- Entry actions: performed when the FSM enters a state.

Although classical models only define entry and input actions, exit and transition actions are also used in practice. On the other hand, whereas input, exit and entry actions are state dependent, transition actions are transition dependent. In addition, every time the FSM changes from one state to another, the four types of actions are performed in the following sequence: input, exit, transition, and entry. Finally, if there is no state change, only the input action may be performed.

Not all these types of actions are usually used for a same system. Formally, if only entry actions are generated, the system is called a Moore model. In contrast, if only input actions are generated, the system is called a Mealy model.

Once the concept of action has been defined, we can return to the representation problem. Using transition matrices and state transition diagrams it is difficult to express the functionality of finite state machines having several actions. This problem could be solved using state transition tables. As can be

State	Entry actions		Entry action 1 Entry action 2
	Exit actions		Exit action 1
	Input actions	Condition 1	Input action 1 Input action 2
		Condition 2	Input action 3
Next state 1	Transition condition 1		
Next state 2	Transition condition 2		

Table 5.4: Example of state transition table

seen in Table 5.4, they contain fields to specify input, exit, and entry actions. Transition actions are not included to keep the table as simple as possible.

Each state has its own state transition table. The first two rows contain the entry and exit actions to be performed when entering and leaving the state, respectively. Next, the table contains all input actions (if any) that are performed when certain conditions are fulfilled over the input variables. Finally, the table may contain transitions to some states. Since it is not possible to make more than one transition at the same time, the sequence in the table will determine the priority of transitions.

5.4.3.3 Our advising model

Before going into details of this section, we would like to thank Dr. Núria Bertomeu Castelló (Bertomeu et al., 2006; Bertomeu Castelló and Benz, 2009; Bertomeu and Benz, 2009; Adolphs et al., 2011; Benz et al., 2011; Bertomeu Castelló, 2012; McDonald et al., 2013) from the Linguistics Department [1] at the University of Potsdam [2] (Germany). This advising model would not have been possible without all her excellent and unceasing work, performed during a one month stay in our research group.

Finite state machines constitute a good control model to manage the global behavior of our system due to factors such as simplicity, comprehension, maintainability, and modularity. In addition, they can take account of the past history,

[1] https://www.ling.uni-potsdam.de/
[2] http://www.uni-potsdam.de/

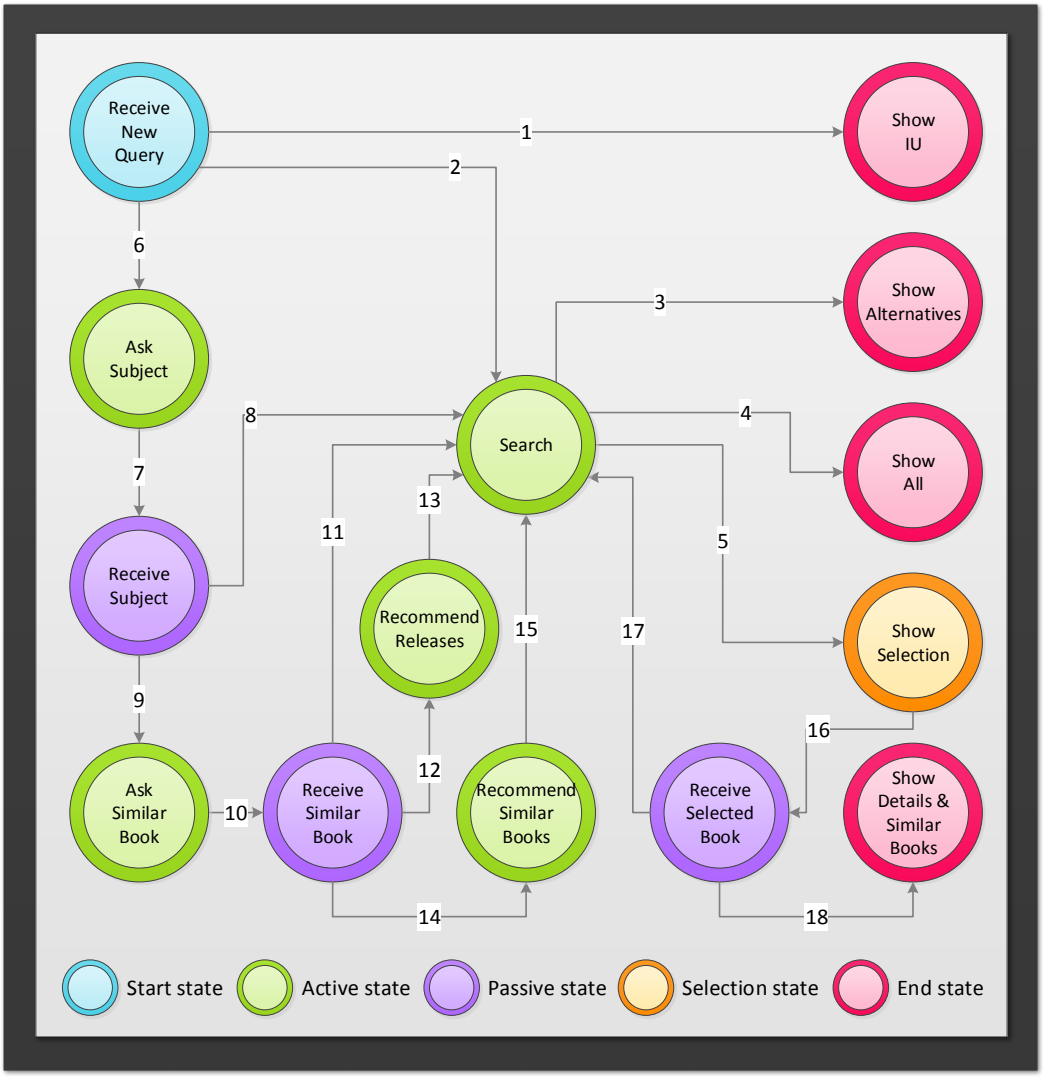


Figure 5.22: State transition diagram of our finite state machine

so the answer not only depends on the last question asked by the user but also the current state of the system. Moreover, they support including business rules to enforce certain strategies (e.g. recommending the books of a given publisher). A general overview of the finite state machine included in our Decision agent can be seen in the state transition diagram of Figure 5.22.

As we can see, we have defined a start state named `ReceiveNewQuery`. This

is the initial state in which the machine is when a new user begins a conversation with the system. Moreover, every time a multi-step process is finished (we will see an example for recommending a book later on), the machine comes back to the start state and it remains ready for a new process to begin.

Additionally to this start feature, we have defined states to be active or passive. Active states do not require any intervention from the user. They are internal states of the machine that have been designed to perform specific actions (e.g. determining which are the best books to `RecommendSimilarBooks`). In contrast, passive states wait for the user to perform an action. For example, selecting a book, refining a previous search by adding more conditions, answering a question asked by the system, starting a completely new query, and so on.

Finally, end states show the user a set of records obtained from the book database. This also happens to the selection state, which can be considered a passive state as well, since it waits for the user to choose a book.

Besides states, the other two elements of our finite state machine are actions and transitions. The set of available actions must be established beforehand because these have to be implemented. However, states and transitions could be dynamically loaded from a configuration file to allow modifying the concrete behavior of the system more easily, although they are also hard-coded at this moment.

Transitions are rules that determine when the finite state machine should move from one state to another. If there are more than one transition for a certain state, these must be placed in order of priority, since only the first that is fulfilled is applied. Transitions are defined on the basis of expressions involving variables and values. Each variable can be set to `NULL`, if no value has been assigned yet (e.g. `CATEGORY = NULL`), `UNKNOWN`, if it has been impossible to determine the value of the variable (e.g. `CATEGORY = UNKNOWN`), or a specific value according to the type of variable (e.g. a string in `CATEGORY = "poetry"`). Note that considering `UNKNOWN` values is really important because if the system asks the user to provide some data (e.g. a `CATEGORY`) and she refuses to give it, we do not want the system to ask it again.

An example of transition rule is shown in Figure 5.23. It states that IF the current state is `ReceiveNewQuery` AND the user has specified a book category

```
STATE == ReceiveNewQuery
∧ CATEGORY != NULL
∧ CATEGORY != UNKNOWN
⇒ STATE = Search
```

Figure 5.23: Example of transition rule

(it is neither `NULL` nor `UNKNOWN`), THEN the finite state machine must move on to the `Search` state to retrieve the books belonging to that category.

Once we know how states and transitions are defined, we will analyze some of the different paths or processes that have been covered by the state transition diagram shown in Figure 5.22. Although the main objective of this finite state machine is advising users and recommending books, new states and transitions could be defined to take other situations into account (e.g. guiding a user throughout the purchase process).

The first path we are going to consider is `ReceiveNewQuery` $\xrightarrow{2}$ `Search` $\xrightarrow{4}$ `ShowAll`. In the `ReceiveNewQuery` start state, the system is waiting for the user to ask a new question. When this is received, two things may happen, as Table 5.5 shows. If the best answer to the question is provided by the Virtual Assistant expert agent, the finite state machine moves on to the `ShowIU` state. On the other hand, if the user specifies a restriction to the Bookseller (e.g. the question "*Romeo and Juliet by Shakespeare*" sets the variables `TITLE = "romeo and juliet"` and `AUTHOR = "shakespeare"`), the machine goes to the `Search` state. As can be seen in Table 5.6, this state includes an entry action to do a direct search over the indexes and the database using the interpretation of the question. This is an active state since it does not require any intervention from the user. Once the search is performed, the system moves on to another state to present the results to the user. In this case, three different transitions may happen depending on how many items are retrieved. If this number is manageable, the machine makes a transition to the `ShowAll` state. Finally, the system goes back the `ReceiveNewQuery` passive state and waits for the user to keep refining her search adding more conditions (e.g. a price range), or make a completely new query.

The path `ReceiveNewQuery` $\xrightarrow{6}$ `AskSubject` $\xrightarrow{7}$ `ReceiveSubject` is much

ReceiveNewQuery	Entry actions	–
	Exit actions	–
	Input actions	–
$\xrightarrow{1}$ ShowIU	ExpertAgent == VirtualAssistant	
$\xrightarrow{2}$ Search	(TITLE != NULL && TITLE != UNDEFINED) ‖ (AUTHOR != NULL && AUTHOR != UNDEFINED) ‖ (PUBLISHER != NULL && PUBLISHER != UNDEFINED) ‖ (CATEGORY != NULL && CATEGORY != UNDEFINED) ‖ (LABEL != NULL && LABEL != UNDEFINED) ‖ (DATE != NULL && DATE != UNDEFINED) ‖ (PRICE != NULL && PRICE != UNDEFINED) ‖ (ISBN != NULL && ISBN != UNDEFINED) ‖ (EAN != NULL && EAN != UNDEFINED)	
$\xrightarrow{6}$ AskSubject	Answer == IU::RecommendationMother	

Table 5.5: State transition table for the ReceiveNewQuery state

Search	Entry actions	Search over the indexes and the database
	Exit actions	–
	Input actions	–
$\xrightarrow{3}$ ShowAlternatives	results.size() == 0	
$\xrightarrow{4}$ ShowAll	results.size() <= 20	
$\xrightarrow{5}$ ShowSelection	results.size() > 20	

Table 5.6: State transition table for the Search state

more interesting. It happens when the user wants the Bookseller to make a recommendation. For example, the query "*I am looking for a present for my mother*" makes the Virtual Assistant generate the information unit IU::RecommendationMother, which makes a transition from ReceiveNewQuery to AskSubject. As we can see in Table 5.7, this new state has an entry action to change the answer to IU::AskSubject. Next, the machine moves on to the ReceiveSubject passive state and waits for the user to make a new input. The answer received by the user in this case is "*Please, tell me what kind of books you*

AskSubject	Entry actions	ANSWER = IU::AskSubject
	Exit actions	–
	Input actions	–
$\xrightarrow{7}$ ReceiveSubject	true	

Table 5.7: State transition table for the AskSubject state

ReceiveSubject	Entry actions	–
	Exit actions	–
	Input actions	–
$\xrightarrow{8}$ Search	(TITLE != NULL && TITLE != UNDEFINED) || (AUTHOR != NULL && AUTHOR != UNDEFINED) || (PUBLISHER != NULL && PUBLISHER != UNDEFINED) || (CATEGORY != NULL && CATEGORY != UNDEFINED) || (LABEL != NULL && LABEL != UNDEFINED) || (DATE != NULL && DATE != UNDEFINED) || (PRICE != NULL && PRICE != UNDEFINED) || (ISBN != NULL && ISBN != UNDEFINED) || (EAN != NULL && EAN != UNDEFINED)	
$\xrightarrow{9}$ AskSimilarBook	Answer == IU::IDoNotKnow	

Table 5.8: State transition table for the ReceiveSubject state

are interested in." (the sentence associated to IU::AskSubject).

In the ReceiveSubject state two different situations have been defined, as can be seen in Table 5.8. If the user specifies a certain subject (e.g. the question "*superheroes*" sets the variable LABEL = "superheroes") or she ignores the Bookseller's request and specifies a different restriction, the machine moves on to the Search state in order to retrieve those books. On the other hand, if the user does not know any specific subject and she expresses it by means of an answer like "*I do not know*", the machine goes to the AskSimilarBook state. There, two entry actions change the answer to IU::AskSimilarBook ("*Tell me some book that she had enjoyed reading.*") and set the variable LABEL = UNKNOWN. Then, the machine moves on to the ReceiveSimilarBook state.

The `ReceiveSimilarBook` state is quite interesting from a dialog management point of view since it is able to completely modify the query to be made to the indexes and the database. For example, if the user answers the question about a similar book using "*The Pillars of the Earth*", the system does not return that book directly but it goes to the `RecommendSimilarBooks` state. This state has an entry action to make a new interpretation to look for similar books. This concept of similarity can be defined as other books written by the same author (e.g. Ken Follett), having some category in common (e.g. historical fiction), and so on.

On the other hand, if the user does not provide a similar book (i.e. the Bookseller is not able to identify a `TITLE` or an `AUTHOR` restriction), the machine moves to the `RecommendReleases` state . This has an entry action to make a new interpretation to look for the most recent books. After that, the machine goes to the `Search` state to retrieve all the information of the books selected by that new interpretation.

5.4.4 The User Interface

As can be seen in Figure 5.24, SmartSeller is placed over the target website in a different layer. The avatar, described in Section 3.3 of Chapter 3, is a 3D model which is able to speak and emotionally react to the questions asked by users. On the other hand, its answer panel contains all the conversation and the recommendations offered by the system to continue the dialog. Finally, in addition to generating dynamic pages with information about the books that match the query as it happens in Figure 5.24, the system can also redirect to certain web pages to support its answers.

5.5 Results

In this section, we will make a comparative analysis and an evaluation in order to confirm that our proposal provides several improvements in the key features that have been described along the chapter.

Figure 5.24: The user interface of SmartSeller for a real Spanish bookstore

5.5.1 Comparative analysis

In Section 5.4 we identified a list with some important features that a good natural language interface (NLI) system should have, considering the broader sense of NLI described in Section 5.1 and depicted in Figure 5.1. The main objective of this chapter was to provide a series of improvements in those key factors. Therefore, those features will be used here to make a comparative analysis between our SmartSeller system and some representative alternatives that were described in Section 5.2. The results of this study have been summarized in Table 5.9, and they will be analyzed along the following paragraphs. All this information has been purely gathered from the papers that describe those systems, with the exception of Anna, a commercial system that has been tested online.

	JUPITER	MERCURY	NLA	PRECISE	ORAKEL	NaLIR	Anna	SmartSeller
2D/3D character	✗	✗	✗	✗	✗	✗	✓ (2D)	✓ (3D)
Advice	✗	✗	✓	✗	✗	✗	✗	✓
Built-in ASR	✓	✓	✓	✗	✗	✗	✗	✗
Built-in TTS	✓	✓	✗	✗	✗	✗	✗	✓
Business intelligence	✗	✗	✓	✗	✗	✗	?	✓
DBMS independence	✓	✓	✓	✓	✓	✓	?	✓
Dialog management	✓	✓	✓	✗	✗	✗	✓	✓
Ease of portability	✓	✓	✓	✓	✓	✓	?	✓
Emotions	✗	✗	✗	✗	✗	✗	✓	✓
Fuzzy queries	✗	✓	✓	✗	✓	✗	✗	✓
General domain queries	✗	✗	✗	✗	✗	✗	✓	✓
Heterogeneous data sources	✗	✗	✗	✗	✗	✗	✓	✓
Linguistic phenomena	✓	✓	✓	✓	✗	✓	✓	✓
Memory management	✓	✓	✓	✗	✗	✗	✓	✓
Multilingual	✓	✗	✗	✗	✗	✗	✓	✓
Temporal queries	✓	✓	✗	✗	?	✗	✗	✓

Legend: JUPITER (Zue et al., 1997; Zue et al., 2000); MERCURY (Seneff et al., 1999; Seneff and Polifroni, 2000; Seneff, 2002); NLA (Natural Language Assistant) (Chai et al., 2002); PRECISE (Popescu et al., 2003); ORAKEL (Cimiano et al., 2008); NaLIR (Li and Jagadish, 2014b; Li and Jagadish, 2014a); Anna [http://www.ikea.com/us/en/]

Table 5.9: Comparative analysis between SmartSeller and other representative systems

2D/3D character When the system is open and it is aimed to be used by the general public, the use of a virtual character may generate more confidence on users and influence their perception of the system. The fact of considering a 3D character instead of a 2D one might contribute to this sensation, although sometimes the difference in quality is not very significant.

Many of the systems included in Table 5.9 lacked a virtual character. JUPITER and MERCURY were two over-the-telephone natural language interfaces, so this lack was justified. Of the other systems, only Anna was represented by a 2D interactive character which allowed engaging with users in conversation in a friendlier way. She could tilt her head, blink, and smile. All the other systems were not represented by any character and the communication was just performed by means of text sentences.

SmartSeller uses the 3D model described in Section 5.4.4 to achieve a much more realistic user experience. It is perfectly integrated with the model of emotions described in Section 5.4.1.4. It can move its head and its eyes, blink, nod, deny, or blush. The opening of the mouth is synchronized with its voice. Moreover, many different accessories can be included (e.g. glasses, hats, or wigs).

Advice Some domains are purely informative and they do not entail a decision making process on the user side (e.g. weather forecast). However, in other domains sometimes the system has to take control of the conversation, so its ability to provide personal advising is really important. Whereas recommendations are offered to the user taking her preferences into account, advice also involves the use of dialog to identify such preferences. Although we consider this one of the most useful features, it has not been usually implemented in the reviewed systems.

Traditional NLIDBs that do not include conversational capabilities such as PRECISE, ORAKEL or NaLIR do not provide this advice feature. In the case of JUPITER, the weather forecast domain is not very suitable, as we have already said. MERCURY could indeed advise users on the process of choosing the best possible flight that matched their needs in terms of price, duration, number of scales, and so on. However, the system just showed the results that were retrieved from the database and it did not provide any special help about decision making. Regarding Anna, although she could recognize when a user asked about a product in a color that was not available, she was not able to give advice in questions like "*I want a toy for a child*", event though that category already existed in IKEA's website and it was divided into several age ranges. From the systems of Table 5.9, just NLA could be said to have some advice capabilities since it redesigned the questions that it asked users to be simpler and focus on usage patterns rather than technical features. This allowed classifying users and applying a different pool of questions in each case.

In contrast to those systems that just focus on product features and only ask users about the attributes on which a restriction has not been imposed yet, SmartSeller asks questions trying to understand the real necessities of the user . As we saw in Section 5.4.3.1, SmartSeller includes an advisor module based on a finite state machine. This allows guiding users when they do not have a clear objective in mind. For example, if somebody wants to buy a present for someone else, and he do not know which book to buy, SmartSeller is able to ask for a book that she had enjoyed and use it to look for similar books for him. On the other hand, this advisor module also helps to make the capabilities of the system evident to users.

Built-in Automatic Speech Recognition (ASR) Nowadays, there are more and more portable devices where traditional input interfaces are limited and not very effective. In addition, accessibility is more and more important. In both cases, automatic speech recognition (ASR) becomes really useful.

Many of the systems included in Table 5.9, like the vast majority of the systems described in Section 5.2, did not include a built-in ASR module. Maybe, the low reliability of ASR technology during many years has contributed to this situation. There are some exceptions such as JUPITER or MERCURY, which provided information over the telephone using spoken dialog, so they included their own speech recognition system (the MIT's SUMMIT (Glass et al., 1996)). In the case of NLA, it supported mixed-initiative dialog with multiple modalities, including typed-in text and speech. Nowadays, it is not very normal that this kind of systems develop their own ASR technology since a great effort is required to achieve as good results as a commercial software.

SmartSeller is in that group of systems that do not include a built-in ASR module. Nevertheless, ASR technology has been significantly improved over the last few years, and nowadays is present in almost all mobile devices, and the results are really impressive. For this reason, the lack of a built-in ASR module is not so important since that functionality could be borrowed from the operating system of the device.

Built-in Text-to-Speech (TTS) As it happens to automatic speech recognition (ASR), accessibility and new portable devices make text-to-speech (TTS) become really useful.

Traditionally, TTS technology has achieved better results than ASR since the former is a generation process and the latter is a recognition process. However, this has not been clearly reflected in the reviewed systems, maybe because TTS is viewed as an accessory feature and in the past the quality of the generated voices (unemotional and canned) was not very good. From the systems of Table 5.9, just JUPITER and MERCURY included a commercial-off-the-shelf solution (DECTalk), since they were over-the-telephone conversational interfaces.

SmartSeller includes its own TTS server with a local cache built on top of

the Nuance Loquendo [1] technology, as we saw in Figure 4.9 of Chapter 4. This speeds up the answer generation process. Nevertheless, most modern operating systems include a TTS engine. Therefore, in most of the cases we should not pay too much attention to this feature when evaluating the quality of a system.

Business intelligence Another very interesting feature is the ability to incorporate business rules into the system logic. Thus, it is possible to reflect the company policies and procedures in the behavior and the decisions made by the conversational system. In many cases, this ability is not provided, that is, the system is not able to modify its answers according to the business requirements. Other times, this behavior is hard-coded, so it cannot be easily changed.

Most of the systems of Table 5.9 simply return the results found in the database and they do not allow prioritizing some of them according to a certain criteria (e.g. higher benefits of the company in some products). If you ask Anna about "*tables*", she does not seem to exhibit any preference when she let you choose from a list of different "product types" sorted by name. However, we do not know if any business rule is applied when search results are sorted by relevance. On the other hand, NLA uses rules to accommodate both customer needs and business requirements. It allows associating rules to certain categories in order to create constraints. For example, a multimedia use category could indicate that a machine should have "a DVD, a high-CPU speed, a large disk drive, and a display with at least 14.1 inches". Probably this definition is a little bit outdated, but the the essence of the concept keeps the same.

In the case of SmartSeller, business rules can also be incorporated into a hierarchy of concepts like the one described in Section 5.4.2.1. Moreover, they can be included into the advice and recommendation process presented in Section 5.4.3.1 (e.g. when the Bookseller recommends similar books or releases). In this way, the user can receive a special treatment that is completely in accordance with the business intelligence of the organization.

[1] http://www.nuance.com/for-business/customer-service-solutions/loquendo-small-business-bundle/index.htm

Database management system independence When developing a generic natural language interface, the independence from the underlying database management system (DBMS) is very important. This allows the system to be used to extract information from any kind of database.

Although in some cases it is not possible to determine the degree of independence, in general most of the systems try to avoid being tied to a specific DBMS. JUPITER obtained forecasts from a relational database using SQL, but this database was completed from multiple websites, so the system had to recognize when two sources provided overlapping information. NLA also maintained a local database that was populated directly from the original databases. This automatic process converted data types and extracted product specifications from multiple attributes on a daily basis. NLA generated constraints that were later translated into SQL statements. ORAKEL was one of the most interesting systems regarding this aspect. It translated questions into logical form, which allowed using an inference engine to provide an answer, even if it was not explicitly contained in the knowledge base but could be inferred from it. Its architecture allowed porting the system to any query language, in particular some RDF query languages or plain SQL to access conventional relational databases. NaLIR also focused on relational databases, although another system from the same author named NaLIX (Li et al., 2005) was able to work with XML files. In the case of Anna, we could not confirm the independence from the underlying DBMS, but the fact of being a commercial system made by an independent company makes us think so.

SmartSeller translates queries into an intermediate representation language using the dynamic semantic grammar described in Section 5.4.2.1. This constitutes an abstract layer that makes restrictions and interpretations independent from the DBMS. In addition, a set of indexes is used to make the system independent from the particular structure of the database of products. Moreover, SmartSeller enriches the database providing an additional classification by means of a hierarchy of labels.

Dialog management When a user interacts with an NLI, sometimes she does not have a clear objective in mind. She can specify an initial set of restrictions

which may change afterwards. For this reason, it is important to include a dialog manager which can engage in conversation with users, warn them if a certain restriction cannot be fulfilled, and so on.

PRECISE, ORAKEL and NaLIR were traditional NLIDBs, that is, they did not include conversational capabilities. In the case of NaLIR, it was interactive, but just in the sense of including menus that allowed changing some parameters of the recognized question. JUPITER and MERCURY were two over-the-telephone conversational systems, so the amount of information that they could provide in a turn had to be very concise. The domain of JUPITER, weather forecast information, did not require extensive dialog management. It was mainly basic information access. However, MERCURY required ten of turns to accomplish a typical task of making a round-trip flight reservation. MERCURY used a mixed-initiative approach and a confirmation strategy when "dangerous" actions were requested to the user (e.g. to avoid an extensive repair process to recover when city name requests were misrecognized and the source city was changed during the itinerary planning stage). Confirmation could become wordy and inefficient, so it had to be used carefully. NLA also proposed a mixed-initiative interaction. Although it gave the initiative to users, the dialog manager asked them specific questions to narrow the recommendation list. It differentiated between two types of users, technology savvy and general, to ask about the specific product attributes or usage patterns that best discriminated among the retrieved products. It also utilized several strategies to deal with different situations: clarification screens with possible queries when no constraints were identified from a user input, valid attribute ranges when the user specified invalid constraints, and so on. A similar strategy was carried out by Anna. For example, at the beginning of the conversation, she encouraged us to tell her our nearest store, or if we looked for a specific product in IKEA and then we specified a color in which the product did not exist, Anna kindly offered us to do a new search with a different color.

SmartSeller is also a mixed-initiative system. The user can take control of the conversation, but when she does not have a clear objective in mind, the system can take the initiative to identify her necessities, as it was explained in Section 5.4.3.1. The feedback that it always provides to the user and the inclusion of delete rules to clear restrictions that we saw in Section 5.4.2.1 avoid having

to interrupt the conversation with annoying clarification dialogs. Nevertheless, what makes SmartSeller different from all the systems of Table 5.9 is that each agent has its own dialog manager. This is an advantage because it allows each agent to adapt itself to the particular features of each problem. For example, the Virtual Assistant agent uses a rule-based system to set a group of priorities and filter the list of recognized information units to decide about which one it is going to talk. On the other hand, the Bookseller agent combines in an intelligent way the different restrictions that have been identified to remove collisions and ambiguity. Thus, it generates all valid interpretations to the question in order to choose the best one according to a scoring measure. Finally, the Decision Agent has a general overview of the system and can perform a unified strategy.

Ease of portability It is the ability to configure or adapt the system to different domains without a significant effort.

During the first years of natural language interfaces, it is true that this posed a serious problem. However, in general, nowadays there is not a really big difference between most of the systems regarding this aspect. Maybe those that perform a statistical parsing or gather some of the domain knowledge automatically have some advantage over the others. In JUPITER, a scripting language controlled the flow through each dialog so that the system could be specialized to a variety of different configurations. NLA allowed scaling to multiple languages and geographies with minimal reconfiguration and its authors developed various tools and processes for the maintenance of the business rules and the statistical parser of the system. In concrete, they maintained a local database to avoid problems with the structure of external databases. PRECISE was able to automatically build the lexicon from the information of the database and WordNet [1] (Miller et al., 1990; Miller, 1995; Fellbaum, 1998), and it was tested on three distinct databases in the domains of restaurants, jobs, and geography. ORAKEL focused on minimizing the effort of adapting the system to a given domain and allowed users which were not familiar with methods from natural language processing or formal linguistics to port the system to a certain domain and knowledge base. NaLIR was used against a number of real application scenarios including Microsoft Academic

[1] http://wordnet.princeton.edu/

Search [1], Yahoo! Movies [2], and DBLP collection [3]. Anyway, if we want to achieve very good results and a great performance, it will always be necessary to adjust or tune the system in order to take account of the distinctive features of each application domain.

In the case of SmartSeller, the fact of being a multi-agent system gives it a certain advantage because we can reuse the different expert and decision agents between applications. In addition, all grammar rules are specified independently, so they can be reused between different domains. Moreover, the possibility of editing the interpretation of fuzzy concepts explained in Section 5.4.2.1 gives each administrator the opportunity of adapting the system to his own necessities. Finally, the use of indexes makes the system independent from the particular structure of the database, as we saw in Section 5.4.2.1.

Emotions One of the problems of conversational systems is their difficulty to emotionally engage in dialog with users. They usually seem cold and with no feelings, and this affects their credibility. Their answers should entail an emotional reaction, which should be reflected both at a textual level of the answer itself and the visual behavior of the agent.

Finding systems that are able to emotionally react during the interaction is not so common. Some of them reflect certain personality or character when they are asked some personal questions. However, it is very superficial because the reaction is only a mere textual output and it does not have an underlying emotional model that could really affect the state of the agent. From the systems of Table 5.9, just Anna was able to show a certain emotional reaction. She smiled with the question "*gift cards*". However, in general her emotional behavior was quite limited and it did not seem to integrate a full emotional model.

If we compare it to the reviewed systems, SmartSeller is, by far, the most complete, natural and realistic system when we talk about emotions. As we saw in Section 5.4.1.4, it includes an emotional state controller based on a dynamic probabilistic fuzzy rule-based system. It defines eight types of emotions and six

[1] http://academic.research.microsoft.com/
[2] http://movies.yahoo.com/
[3] http://dblp.uni-trier.de/

different personalities, and they are visually reflected by means of a 3D model. Emotions can completely change answers, up to the point that they could make the system deny answering any other question until the user apologized if he has had a lack of respect to it. On the other hand, personalities modify the influence that events have on the system.

Fuzzy queries People do not always use a perfectly accurate language to express themselves, but they often do it in an imprecise or incomplete way. Although this is something natural for humans, it entails an extra level of complexity in human-computer interaction by means of natural language. For this reason, it is really important for the system to be able to recognize imprecise queries that contain fuzzy terms.

MERCURY used a fuzzy matching heuristic to understand users when they employed inexact terms to refer to departure/arrival times, airline names, flight numbers, and source/destination/connection cities. For example, a flight departing at 7:56 AM might be referred to as "*the eight o'clock flight*". NLA mapped qualitative constraints (low or high constraints on numeric product attributes) to specific constraints based on automatic partitioning of the current range of values. For example, considering high the top one-fifth of the capacities of that time, "HDD size: high" was mapped to "HDD size >20 GB". NLA also allowed queries containing terms such as "*cutting edge*". ORAKEL expected a definition of the semantics of an adjective in terms of a rule, and it could only handle scalar adjectives such as "*big*", "*high*", "*long*", etc. All the other systems did not handle fuzzy queries. For example, although NaLIR could handle complex queries containing aggregations, comparisons, and various types of joins and nesting, it was not designed to handle imprecise terms. The same happened to Anna, who could not correctly recognize questions containing fuzzy terms such as "*cheap*" or "*small*".

SmartSeller not only recognizes fuzzy queries, but it also allows configuring how these queries are interpreted, as we saw in Section 5.4.2.1. For example, the administrator of the system can modify the price range for what is considered cheap, or the date range that makes something be recent, since these values can be significantly different for two bookstores that sell different kinds of books.

General domain queries It has always been interesting for any conversational system to include enough knowledge to answer general purpose questions about the personal information of the avatar, its hobbies, and so on. This is something that appeals to users and entertains them, and it is able to arouse their curiosity to see how far the system could get. For this reason, a large number of questions are always related to this matter. However, if we focus excessively on this type of questions, we run the risk of making users think that the system has been designed for that purpose instead of being a natural language interface. In addition, the fact of giving a bad answer to any of that out-of-domain questions might negatively influence the perception that users have of the system.

Most of the systems included in Table 5.9 were traditional NLIDBs that focused exclusively on the access to the information stored in the database, and they completely forgot about general domain questions. Maybe it is not so important to answer those questions properly, or at least how the user would expect, but the fact of being able to recognize those questions and use the answers to reorient the conversation towards the objective for which the system has been designed (e.g. product selling). This is what some systems like Anna do. For example, she tried to relate intimate questions with specific products of IKEA.

SmartSeller shares this idea of reorienting the conversation, but it also contributes a different point of view in the sense that it uses several expert agents that allow making use of different data sources. For example, the Virtual Assistant uses a hierarchy of ontologies that not only indexes the website to answer frequently asked questions about the shop (e.g. "*opening hours*", "*payment methods*", or "*delivery charges*") but it also allows answering questions about the weather, the time, or the personal information of the assistant. In addition, we could include new expert agents to extend that functionality. For instance, an agent to query Wikipedia [1] or DBpedia [2] (Auer et al., 2007) would suppose a huge source of knowledge.

Heterogeneous data sources Information is not stored in just one place but it is scattered across different sources. There are innumerable services on the

[1] https://www.wikipedia.org/
[2] http://www.dbpedia.org/

Internet that offer very specific information and of great quality. Making one of those systems from scratch could be really expensive. For this reason, services like Apple's Siri [1] use third-parties to offer certain contents (e.g. Wolfram Alpha [2] for mathematical questions, or Yelp! [3] for relevant business near you). This confirms the importance of integrating heterogeneous data sources that could be accessed by the system at the same time. Sometimes the information will be stored in a certain location (e.g. a product database), and others in another different place (e.g. a website). However, this must be completely transparent to users.

As we can see on Table 5.9, most of the systems were not designed to work with different types of data sources at the same time. JUPITER just handled one type of content, weather forecast information that was integrated into a local database, although it is true that that information was gathered from multiple websites and this entailed additional problems. In MERCURY, the information on flight schedules and pricing was obtained in real-time from the Sabre Group's Travelocity service, although it was complemented with a small local database of pragmatic information, such as geographic location of airports. ORAKEL was DBMS independent, but it was not designed to work with heterogeneous data sources simultaneously. NaLIR was an interactive NLI for querying relational databases, and although a similar system from the same author, NaLIX (Li et al., 2005), could query XML files, they were independent systems that did not communicate themselves to resolve a same problem collaboratively. There seemed to be just one exception, Anna, who could not only answer questions about the product database, but also index (probably by hand) certain sections of the IKEA's website (e.g. "*return policy*"). However, this ability was rather limited and it was not as versatile as the one proposed by SmartSeller.

The architecture and multi-agent nature of SmartSeller suppose a great advance since they allow the user to access the information independently of being stored in different locations or using diverse formats. As we saw in Figure 5.1, SmartSeller could interact not only with databases or websites as it happens to

[1] https://www.apple.com/ios/siri/
[2] http://www.wolframalpha.com/
[3] http://www.yelp.com/

most of the systems included in Table 5.9, but also any kind of data source like ontologies, social networks, XML or plain text files, forums, APIs, and so on. Of course, provided that an expert agent has been made to handle that type of knowledge, in the same way as an operating system needs different drivers to communicate with distinct devices.

Linguistic phenomena Making a good conversational system is rather complex due to the richness of natural language. There are many linguistic phenomena that could appear: anaphora (the use of an expression the interpretation of which depends upon another expression in context), ellipsis (the omission of one or more words that are nevertheless understood in the context of the remaining elements), ambiguity, incomplete search values, and so on. Handling as most of these phenomena as possible is very important to perform a good recognition.

JUPITER, like most of the systems, allowed unimportant words to be skipped. MERCURY developed a fuzzy matching heuristic that accounted for departure and arrival times, airline names, flight numbers, and cities. It also handled ambiguous intentions to book a flight. For example, it was unclear whether "*the later flight*" meant that the user wanted more information or she wanted an actual booking. In that case, MERCURY adopted the general strategy of treating those potential booking requests as more information requests and followed up with "*Shall I book this flight for you?*". NLA handled incomplete search values matching the expression in a particular attribute category with values of that attribute in the product database and choosing the most similar one (e.g. the constraint `OS` = "*win xp*" was matched to `OS` = "*Windows XP Professional*"). PRECISE was able to handle free-standing values, where the attribute was implicit (e.g. in "*What French restaurants are located downtown?*", the word "*French*" referred to the value `French` of the implicit database attribute `Cuisine`). For string matching, it saved words and their WordNet synonyms in a hash table so that it could convert each word into the original value in the database. In case of ambiguity, PRECISE simply asked the user to choose between the distinct semantic interpretations. ORAKEL assumed a full parse of the input sentence and thus expected the sentence to be grammatical. However, if the question was not grammatical it simply failed and told the user that it did not understand the

question. This behavior is unacceptable since handling ungrammatical input and unknown words is a must for any 'commercial' natural language interface. NaLIR extended the generated parse tree to avoid that some words failed in mapping to database elements due to the vocabulary restriction of the system. Regarding ambiguous interpretations, it always showed multiple options for the user to choose from. This way of handling ambiguous interpretations was also used by Anna. She let you choose from a set of different options (e.g. product types, product names, colors, or price ranges).

Regarding SmartSeller, it employs several approaches to handle the problems that have been mentioned. On the one hand, the Virtual Assistant uses a token-based recognition algorithm that allows extracting information from questions even though the system is used like a search engine and many words are omitted. The ambiguity in this case is resolved by means of a rule-based system which assigns different priorities to the recognized information units. In addition, it can use the guided tours that were mentioned in Section 5.4.1.1 to implement decision trees. On the other hand, as we presented in Section 5.4.2.1, the Bookseller uses a dynamic semantic grammar which allows detailing the structure of sentences up to point that is needed in each specific case. All of this without having to overanalyze the sentence as it happens when using a syntactic parsing. This fine-grained control is used in delete conditions, where the Bookseller takes advantage of certain words that are usually omitted by most of the other systems, but which are essential to understand the meaning of the sentence (e.g. the word "*any*" in "*any author*" suggests to clear a previously imposed `AUTHOR` condition). The Bookseller also handles the use of incomplete search values as it was seen in Section 5.4.2.2. Moreover, it has its own mechanism for resolving ambiguity. In Section 5.4.2.3 we saw how it combines restrictions in an intelligent way to generate a list of possible interpretations that are ranked according to a certain score.

Memory management During a dialog, people often omit certain words that can be perfectly understood according to the situation. In a conversational system, the use of the memory allows putting the dialog in context. In this way, the user can refer to some concepts implicitly. On the other hand, when users

interact with a natural language interface, sometimes they do not have a clear objective in mind, but that idea takes shape as the conversation flows. Initial restrictions are modified, new ones are defined, etc. For this reason, it is very important for the system to have a memory that can record any piece of useful information that can be found in the conversation.

Some systems like PRECISE and ORAKEL were not interactive and questions had to be self-contained. They could not reuse any past restriction because they did not have a memory to remember things. NaLIR showed a "query history", but it just contained a list with the questions asked by the user. JUPITER did take account of the prior dialog context for processing user queries. For example, if the system listed a set of cities in California, during the recognition it preferred hypotheses that contained one of the cities on that list. MERCURY permitted users to modify previous legs by specifying updated constraints (e.g. "*I want to arrive in Tokyo a little earlier*"). NLA maintained a session context and integrated the constraints identified from the last input with the ones captured previously in the session. It also provided feedback on what constraints had been understood to that point. Finally, Anna could store objects, which could be used to answer questions about some of their properties without having to specify their names again. For example, if we asked "*black ADDE chair*", and then "*how much does it cost?*", Anna understood that the latter question referred to the object that was mentioned in the first question. The same happened if we specified a product and a color, and then we changed the color.

SmartSeller can also store different kinds of information units in memory. As we saw in Section 5.4.1.1, the Virtual Assistant uses an ontology to distinguish between objects, properties, specific subjects, general subjects, subject lists, and guided tours. Although this set is broader than those of the reviewed systems, the concept is essentially the same. What makes SmartSeller special is that we can specify different rules to control the way each expert agent reuses the memory. For example, a rule states that if a user asks a question that includes a `PRICE` restriction, the Bookseller may later transform it into an `ORDER BY` condition instead of throwing it away if it cannot be fulfilled when new restrictions arrive. Additionally, in Section 5.4.2.1 we saw how the Bookseller includes delete rules which allow the user to clear specific restrictions from memory without having to

start a completely new query.

Multilingual Internet is not only a mass medium but also a global medium. For this reason, being able to offer the information in different languages is essential when designing a conversational system that is aimed to the general public.

Although some systems like NLA stated that they could be easily scaled to support multiple languages, the reality is that many of them were only tested in one language. This is the case of MERCURY, NLA, PRECISE, ORAKEL and NaLIR, which were available only in English. The process of making a natural language interface multilingual is not always trivial. The use of some techniques can help to reduce the effort that is necessary. For example, the statistical parser should enable NLA to scale to multiple languages and domains in a more robust and reliable way. They stated that to create a French version, they only needed to collect a corpus of French sentences and annotate them with the existing schemes, instead of recruiting French-speaking linguists to create rules for French expressions. JUPITER was originally in English, although the authors started to conduct research on multilingual conversational interfaces, including German, Japanese, Mandarin Chinese and Spanish. At first, they just seemed to generate German reports and they did not handle input queries in these languages, but then they developed an end-to-end Japanese version named MOKUSEI (Nakano et al., 2001).

If we only took the number of languages into account, Anna would clearly stand out over the others since it was available in 16 different languages (English, Chinese, Czech, Danish, Dutch, Finnish, French, German, Hungarian, Italian, Japanese, Norwegian, Portuguese, Russian, Slovak and Spanish). Nevertheless, it should also be considered how easy is to incorporate a new language into the system. Unfortunately, measuring this in an objective way is a difficult task.

SmartSeller has been designed to share the knowledge structure amongst all languages in order to avoid duplicities and reduce the maintenance costs. The system has been successfully tested both in English and Spanish, and the Virtual Assistant expert agent has also been tested in Portuguese. Although the language can be changed during the conversation, at this moment the system does not include a language detection mechanism, so the user must change it manually by

means of the interface.

Temporal queries When a conversational system or a natural language interface is aimed to search items in a catalog, it is perfectly normal for users to make temporal queries, that is, questions that involve relative dates and might retrieve different results depending on when they are asked.

Most systems are not able to handle temporal queries and, at the most, they just sort the results or filter them by a certain range of dates explicitly specified in the question. However, they cannot resolve relative dates such as "*this month*" or "*last decade*". For example, Anna was not able to recognize questions like "*top ten tables 2014*" or "*best selling tables 2014*". In the former case, she understood "*table top-angled*". In the latter, she just recognized the word "*table*". JUPITER did recognize temporal terms such as "*today*" or "*tomorrow*". This could entail additional considerations. For example, calls after midnight asking for "tomorrow's" weather really wanted "today's" weather, defined from midnight to midnight. In MERCURY, the interpretation of dates was also essential for the proper recognition of questions (e.g. "*I want to fly from Boston to London on British Air next Friday*"). In the case of ORAKEL, this feature could not be confirmed, although it might be present because the use of a foundational ontology DOLCE provided predicates and relations related to time and space, which were crucial for representing the semantics of spatial or temporal prepositions.

SmartSeller is not only able to answer questions regarding advance date filtering and sorting (e.g. "*poetry books from the 50's*" or "*the last book of Suzanne Collins*"), but as we saw in Section 5.4.2.1, the grammar of the Bookseller also allows resolving temporal queries such as "*the best book of last year for children*" or "*ten most popular books of the month*". This is a very powerful feature since users could get a general overview of the catalog by means of summaries.

5.5.2 Evaluation

In this section, we will present some details about the implementation and usage of SmartSeller.

The ontologies of the Virtual Assistant expert agent contains around 140 in-

formation units (IUs). The 38% of them are related to the bookstore, whereas the other 62% correspond to general knowledge and information about the avatar's personality. These IUs are distributed in objects (21%), properties (29%), specific subjects (10%), general subjects (38%), and subject lists (1%).

The current version of the grammar of the Bookseller agent has approximately 150 semantic rules containing around 500 expressions (i.e. right parts). These rules are organized in four different categories: restrictive rules (52%), sorting rules (15%), top rules (18%), and delete rules (15%). Top level rules (i.e. ∼`CONDITION`) represent the 25% of all rules.

Most of the rules are restrictive because these include all the attributes of the database that can be used in the search process (title, author, publisher, category, label, date, price, ISBN, and EAN). On the other hand, at this moment sorting rules are only available for the most common attributes (date and price). However, the effort required to include new rules to allow sorting by other attributes is minimum. Moreover, as we said in Section 5.4.2.1, this set of rules is dynamically extended with specific rules that depend on the concrete question asked by the user.

The first version of the grammar was made in few days of development. It just contained the essential rules that allowed capturing the values of the different attributes of the database. In this way, we could have a completely functional system from the very first moment. However, when the first tests were carried out, we realized that this reduced set of rules generated a lot of ambiguity. Some of the questions asked by users contained common expressions that were not being produced by the grammar rules that we had made, but they were being captured by the dynamic rules that were added to the grammar automatically. In particular, these expressions partially matched some book titles generating undesirable restrictions. For this reason, we had to add more expressions, some complex rules and restriction filters during the following weeks, as users continued interacting with the system.

The effort necessary to adapt the system to different languages or application domains depends on each particular case. In general, it could be considered low or medium. Although each case has its own peculiarities, many grammar rules (e.g. those about dates, numbers, or prices) can be reused, and the system can perform

well with a relatively small number of rules. Obviously, the more complete the grammar is, the better for the recognition since less ambiguity is generated. On the other hand, all specific grammar rules are made dynamically cleansing the information from the database, so new domains might require to adapt those filters or implement new ones.

Regarding the usage of SmartSeller, although it is not publicly available yet, a reduced number of people (around 40) have interacted with it so far. Therefore, the following results should be taken carefully since they might be biased.

Users have asked approximately 500 questions along 200 sessions. These questions have an average length of 3.5 words. If we only consider those answered by the Virtual Assistant, the average falls to 2.4 words, whereas it reaches the value 4.3 for those answered by the Bookseller. Questions about books are usually larger because they contain some information about the book the user is looking for (e.g. the title).

If we focus on the behavior of the multi-agent system, the 40.73% of these questions have been answered by the Virtual Assistant agent, whereas the other 59.27% have been handled by the Bookseller. After having performed a deep analysis, we can say that the 93.95% of all questions have been answered by the right agent. If we consider the remaining 6.05%, many of the errors (80%) are questions that had not been initially introduced in the knowledge of the Virtual Assistant (e.g. "*I would like a refund*") and they partially matched the titles of some books.

Considering the Virtual Assistant agent, the 86.63% of the questions have been properly answered. These are related to the bookstore (e.g. "*What's your phone number?*", "*How could I buy a book?*"), the private life of the avatar, greetings, compliments, rude words... Regarding misunderstood questions, they are mainly about topics not covered by the assistant (e.g. "*after-sales service*") and some out-of-domain questions not related to the bookstore.

On the other hand, the Bookseller agent has answered the 84.69% of its questions successfully. Interpretations are made of 1.62 restrictions on average, which specify book titles, authors, publishers, dates, prices, and so on. This makes evident the importance of having a good memory management module since restrictions are usually specified by users little by little. In this case, some errors

have been caused by categories which are not defined in the book database and we had not included in our extended label hierarchy (e.g. "*adventure books*"), so they partially matched some titles (not so bad, but not so good either). In addition, some questions asked the Bookseller for extra advising capabilities (e.g. "*I want a book which doesn't make me think*"). This reflects the necessity of improving the advising module of the Decision agent.

5.6 Conclusion and future work

In this chapter we have presented a generic multi-agent architecture for conversational systems that allows accessing heterogeneous data sources, the kind of problems that can be usually found nowadays in which the information may be scattered across different origins, each one with a particular structure and format. This architecture has been used to make SmartSeller, a specific system for the e-commerce portal of a bookstore.

Our system provides a great improvement regarding others if we focus on the broader concept of natural language interfaces (NLIs) that can be extensively used by any kind of public and not only expert users or database administrators. The majority of the systems have just focused on a very specific part of the problem. PRECISE, ORAKEL and NaLIR had interesting features as NLIs. For example, ORAKEL was focused on portability issues, and NaLIR supported complex SQL statements containing comparisons, quantifications, aggregations, nesting, sorting and various kinds of joins. However, they used just one knowledge source and lacked conversational capabilities. On the other hand, JUPITER, MERCURY and NLA did include a dialog manager, but they neither supported general domain queries nor were able to work with different types of data sources at the same time. In the case of a commercial system like Anna, it was much more complete in the conversational side, but not so advanced in the interaction with the database (e.g. it did not handle fuzzy or temporal queries), and it did not seem to include a full emotional model. Furthermore, most of the systems did not allow advising users when they did not have a clear objective in mind, and they did not incorporate business rules into the dialog management and recommendation processes.

Regarding the future work, some aspects of the system related to the errors

mentioned in Section 5.5.2 should be improved. For example, to solve the issue of the categories that are missing from the book database, we could apply data mining techniques to the descriptions of the books in order to augment our label hierarchy automatically by means of n-grams. We should also define new fuzzy labels to be able to answer some of the questions asked by users (e.g. "*I want a book which doesn't make me think*"). In addition, this extended label hierarchy would become really useful when making recommendations.

Moreover, it would be necessary to make new grammar rules to answer more types of questions directly. For example, specific questions about a certain book (e.g. "*Who wrote...?*", "*How much is...?*"...).

It would also be very interesting to improve the Decision agent with new advising capabilities. For example, new states could be defined to request the user to provide a subtopic when a category is quite big and contains many books (e.g. "*What kind of history books are you interested in? World, Europe, Americas, Ancient Civilizations...*").

In addition, for those cases in which a great amount of records are retrieved from the database, we could implement an algorithm that allowed classifying the search results and sorting them based on diversity and representativeness. In this way, instead of showing all the results sorted by a certain field such as the date or the price, we could make a much more suitable selection.

On the other hand, as we said in Section 5.4.2.3, although for each question we have just focused on the best interpretation found by the Bookseller, the system could be improved if the top 3 interpretations were combined and showed to the user when the best one matches only loosely the sentence.

Additionally, although choosing the answer provided by just one of the expert agents is enough most of the times, sometimes it could be interesting to integrate the answers provided by several agents.

Another issue arises when it is not possible to retrieve any record from the database that can fulfill all the requirements specified by the user. In these cases, the system starts to relax restrictions according to their importance level until it can return some result. We could improve this algorithm so that, instead of having to always remove restrictions, it could replace them with alternatives based on similarity (e.g. an author of the same style, a similar subject, and so

on).

We could also improve the recommendation algorithm used by the advising module. Instead of basing it exclusively on content, we could use a hybrid approach that added a collaborative component. Thus, it would not only take advantage of the structure and information contained in the records of the database, but also of the preferences of users with similar features.

Finally, with regard to the out-of-domain questions, the general knowledge of the system would be greatly improved if we added new expert agents to query external resources such as Wikipedia or DBpedia (Auer et al., 2007).

This page intentionally left blank

6

Conclusions and future work

In this thesis we have studied three problems related to conversational systems such as the improvement of their naturalness, the information overload, and the access to heterogeneous data sources. Along this chapter we will analyze which objectives from the ones posed at the beginning of the thesis have been reached and we will make a series of conclusions. Moreover, we will establish some possible lines for future work.

6.1 Conclusions

By means of this thesis it has been made an analysis of different proposals of literature about three interesting specific problems related to the field of conversational systems: the improvement of their naturalness, the information overload, and the access to heterogeneous data sources. This has allowed detecting some lacks existing in these systems. Based on them, a series of indispensable requirements has been set for solving each of the problems that we faced. These

requirements have allowed us to make concrete value propositions, in the shape of generic models, which we will talk about next. These designs have been used as the basis in the development of specific systems to solve real problems and they are commercially available.

In general, we can say that the objectives that we set at the beginning of this thesis have been successfully achieved and each of our hypotheses have been confirmed:

- The application of fuzzy rule-based systems has allowed designing an emotion control model for the improvement of the naturalness of conversational agents which is easy to interpret and modify. The results obtained in Section 3.6 of Chapter 3 and a series of publications in international journals and congresses (Eisman et al., 2009a; López et al., 2013; Castro et al., 2007b; Váquez Granado et al., 2008; López et al., 2008; Castro et al., 2010a; Castro et al., 2010b) reflect the quality of our proposal.

- The organization of the contents of the website of a certain organization by means of semantic structures such as ontologies, and the application of rules and restrictions at the levels of language recognition and dialog management as well as answer generation, have allowed making a system that takes advantage of all the existing knowledge and makes easy the immediate and precise access to great amounts of dynamic information. The analysis of results shown in Section 4.5 of Chapter 4 and several publications (Eisman et al., 2012; Eisman et al., 2009b) support this work.

- The use of a multi-agent approach based on a set of expert agents coordinated by a series of decision agents has allowed providing a modular and scalable solution to the problem of the access to heterogeneous data sources in an integrated, transparent, effective, efficient, and pleasant way. The comparative analysis and the evaluation performed in Section 5.5 of Chapter 5 and some articles (Eisman and Castro, 2014; Eisman et al., 2015 —*to be published*—) reflect the utility of the system developed.

Next, we detail to what extent and how these general objectives have been accomplished in each of the three problems tackled.

6.1.1 Emotions in Embodied Conversational Agents

Regarding the first of the problems, the one of the improvement of the naturalness of conversational systems or agents, a research into different emotion representation and control models was carried out. However, projects like the Virtual Simulated Patient imposed a series of requirements that existing models were not able to satisfy. That is why it was necessary to design a new control system which has improved some components already included in some models and has included a series of innovatory features:

(a) It allows showing eight different types of basic emotions according to eight emotional attributes: joy, disdain, anger, fear, worry, surprise, sadness, and embarrassment. These attributes are enough for the majority of applications, although the model can be easily extended to show a larger number of emotions if necessary.

(b) It defines six different types of personalities based on the eight emotional attributes: normal, anguished, depressive, hypochondriac, maniac, and phobic. These influence the way the agent perceives its environment. As it happens with basic emotions, these can be extended to cover the particular necessities of each problem.

(c) Each basic emotion is gradually defined in a real scale from 0 to 1, so it can be shown at different intensity levels. A value of 0 means an absence of this emotion, whereas a value of 1 means a total or maximum presence.

(d) The control of the emotional state of the agent is carried out by means of a set of fuzzy rules based systems, one for each of the eight emotional attributes. This allows that it can be specified, with the help of experts at each domain, a series of updating rules for each attribute. Each rule is made of a set of antecedents and a consequent. Antecedents can be related to variables such as the personality of the agent, its health conditions, its environment, the type of question made by the user, or the memory. On the other hand, the consequent tells the degree of variation to apply on the corresponding emotional attribute when the rule is fired (high/medium/low

negative, zero, low/medium/high positive). Compared to other type of approaches, the use of rules allows perfectly understanding and justifying the behavior of the system.

(e) By means of the rules, it is possible to configure how certain factors such as the personality or the emotional state of the agent influence. For example, in the case of the Virtual Simulated Patient, it can be controlled how it perceives its environment, its health conditions, or the illnesses it suffers from.

(f) Each answer can be associated to a certain emotional state. Thus, this will be expressed only when the agent is really in that state. This avoids canning answers which lack any kind of emotional component.

(g) The stability in the emotional state of the agent is got by means of two types of rules. On the one hand, those that do not vary the emotional attributes. Since they have a null variation, when fired they decrease the impact of the rest of rules. On the other hand, the rules that vary the emotional attributes according to the personality make the agent tend to an equilibrium state.

(h) The graphical representation of the conversational agent was initially made by means of an easier 2D model which simplified a little all the emotion management process. Later, this has been replaced by a more advanced 3D model which allows a higher degree of realism, the combination of basic emotions, and improves the transition between different types of emotions.

(i) The non-deterministic behavior of the emotional state of the agent is assured associating a real value between 0 and 1 to each updating rule. This is equivalent to the probability of considering that rule in the current iteration of the system. In addition, these probabilities can be updated dynamically as the conversation goes on.

In order to measure the accuracy of the developed emotional state control system, some interviews (complete dialogs) to a virtual simulated patient were taken. After being checked by domain experts, these determined that, in general,

the behavior shown by the agent happened to be quite natural. In 71.8% of the cases, the generated emotional behavior was the expected one, whereas in the remaining 28.2% some little variations in the emotional attributes opposed to the expected ones were detected. Although it is true that it was necessary to fix the particular probability values associated to some rules at first, this ensured that the system would behave in a non-deterministic way.

In short, the contributions made have allowed improving the naturalness of the conversational systems and hence their credibility. In some applications, like the one of the Virtual Simulated Patient, this can become an essential aspect since the success in the interaction is partly determined by the degree of realism that could be achieved. In other cases, it is quite advisable since it significantly improves user experience, making all the process much more natural. For example, it can encourage a user to make a purchase in an e-commerce portal.

6.1.2 Closed-domain virtual assistants

Second of all, it was studied the problem of information overload, which usually generates certain disorganization and scattering in websites. There were analyzed some of the existing models for navigation improvement. These presented some lacks for their application in a domain like the one of the University of Granada. For this reason, a methodology and a framework were developed for the design of closed-domain virtual assistants. These would allow accessing a great amount of information in an immediate and precise way, including all the features that our problem required:

(a) The design of different types of information units (objects, properties, specific subjects, general subjects, subject lists, and guided tours) allows taking advantage of the specific knowledge provided by the domain. Thus, it is possible to answer questions precisely, or more generally, depending on what it is necessary.

(b) Information units can be organized hierarchically following the same philosophy and principles of object oriented programming.

(c) Answers can be adapted dynamically according to a series of restrictions. In this way, it is possible to offer different information for a same concept depending on the date or the time the question is asked (useful for giving information about summer/winter opening hours, or deadlines, for instance), if it has been previously asked or it is the first time, and so on.

(d) The assistant works independently to the web and can give information that is complementary to the same. Sometimes, it can answer using summarized contents of the web. Other times, it can provide information that is not even included in it, but in a downloadable resource or a link to an external website, for example.

(e) The knowledge about the different information units about which the assistant can inform is stored in a series of ontologies organized hierarchically. This structure allows defining independent ontologies which make easy the reuse of knowledge in different domains and are able to answer general domain questions.

(f) The behavior of the assistant when choosing an answer can be configured by means of a rule based system. This allows altering the priorities assigned to each type of information unit (IU), and consequently the answer to provide, according to a set of parameters such as the class the recognized IU belongs to, the current emotional state of the agent, its personality, or the last answer given by the system, among others.

(g) The memory module and the way information units are defined and organized in the ontology allow placing the conversation in a specific context and reusing the objects that have been mentioned. This allows omitting implicit words during the dialog.

(h) Information units, as well as being organized hierarchically in the ontology, they can be connected with each other making a directed graph. All these links allow making recommendations based on the context of the conversation.

(i) To increase the efficiency of the system and make it able to answer in real time to a larger number of users simultaneously, the knowledge necessary for the questions recognition and the answers generation is preloaded in memory and its access is made by means of a set of indexes.

(j) The assistant can be configured to understand and answer questions asked in any language. All of them can share the same knowledge structure to make the maintenance easy. Until now, it has been successfully tested in three different languages: Spanish, English, and Portuguese.

(k) At present, the assistant does not have its own automatic speech recognition module. However, this functionality can be obtained from the operating system in which the user utilizes the application. What has been developed indeed is a speech synthesis web service based on a commercial software. This has allowed making a sound file cache that decreases response times.

(l) Through the ontology and by means of the use of rules, it is possible to design processes specifying the different stages or phases they are made of. In this way, the assistant can guide users through those paths that have been previously defined.

(m) The restricted access to information is achieved thanks to the use of rules and the specification of restrictions in answers. In this way, it is possible for the assistant to deny the user the access to particular resources or ask her to identify herself previously. This does not imply that the website does not have to implement the necessary security measures, since the assistant would only be redirecting the user.

(n) Taking advantage of the definition of the different classes of objects existing in the ontology, with their specific properties or attributes, it is possible to automatically generate outlines which offer all the information related to a certain object in one go.

(o) Rules and restrictions are perfectly compatible with the emotional state control system. Thus, it is possible to modify the particular linguistic

expressions used when answering each question. Even, it is possible to completely modify the concept about which it is going to be informed.

This methodology and the proposed system have been used as the basis in the creation of different closed-domain assistants such as the ones of the University of Granada (Spain) or the University of Beira Interior (Portugal). Considering the former, we can say that the assistant has answered more than 500,000 questions from 167 different countries throughout five years and a half, since 2010. In this case, the recognition rate is equal to 94.39% for in-domain questions (those for which the assistant has been designed), or 76.30% if we also consider out-of-domain questions (about her private life, her hobbies...), which represent the 34.41% of total questions.

To sum up, the contributions made in this field of closed-domain virtual assistants have allowed developing the first virtual assistant of a Spanish university, a problem that included some particular features: great amount of information spread over different independent subdomains; existence of various user groups with different interests (teaching and research academic staff, administrative staff, or students); dynamic information which varies throughout the time; presence of external resources (regulations, decisions of official organizations...); possibility of informing in several languages or at different detail levels, etcetera.

6.1.3 Accessing heterogeneous data sources

The last of the analyzed aspects has to do with the kind of problems that can be usually found nowadays. The usual is that the information is scattered across different sources which must be integrated in a simultaneous and transparent way for the user. However, instead of focusing on the broader concept of natural language interface that can be used by any kind of public, most of the existing systems focus on just one very specific part of the problem. That is why it was necessary to design a new generic architecture which has been used to make SmartSeller, a specific system for the e-commerce portal of a bookstore. This architecture provides numerous advantages with regard to others:

(a) The integration of the different data sources is carried out by means of

multi-agent system made of two types of agents. There are independent expert agents which are specialized in the access to different knowledge sources. These are coordinated by a series of decision agents to provide a coherent final answer to the user.

(b) Although the recognition using tokens can be enough for certain domains or applications, these do not help to understand the question precisely since a lot of information is lost. However, the use of semantic grammars, as well as reducing the elliptical problems that can cause some queries, it allows us to make the most of all the available information. Thus, it is possible to extract all possible interpretations from each question and determine the best based on a score mechanism.

(c) To improve the accuracy of the system, the grammar is dynamically extended with a set of production rules generated automatically from the information stored in the database. In addition, applying a series of pruning algorithms the performance of the grammar is improved.d

(d) The system includes a set of indexes over the database which allow processing the information to cleanse and filter it, increasing thus the reliability of the recognition and speeding up all the process.

(e) The memory and the dialog management modules allow the reuse of restrictions along a same conversation. In this way, the user need not specify all restrictions at once, but she can do it little by little. Which components to reuse and how to do it is determined by a rule-based system.

(f) The advising module uses a state machine to represent all possible situations and stages of the relation between the user and the system (e.g. receive query, ask subject, ask similar book, recommend releases, recommend similar books...). In addition, it is possible to associate to each state a series of actions that must be carried out when entering or existing it. Thus, when for example the user does not know very well what to look for, the system will be able to advise her asking for books that she had enjoyed.

(g) The use of taxonomies or ontologies allow setting a category hierarchy which enriches the information existing in the database. In this way, the system can go beyond and answer questions that could not be answered considering only the information stored in the database. In addition, these hierarchies can be reused in different application domains.

(h) The grammar includes production rules which allow recognizing fuzzy concepts such as "*cheap*" or "*recent*". The way these concepts are interpreted can be configured to adapt them to each particular domain.

(i) By means of a series of dynamic production rules, the grammar also allows recognizing and interpreting the relative dates included in temporal queries. In this type of questions, the information provided will depend on when the query is made.

(j) The use of a multi-agent architecture which follows a divide and conquer philosophy allows the reuse of different knowledge sources. Moreover, the existence of generic production rules (e.g. the related to dates or prices) and the extension of the grammar by means of dynamic rules, make easy the adaptation of the system to different domains.

(k) The system has been designed for being able to include the information regarding different languages taking advantage of a common knowledge structure. Until now, it has been successfully tested in Spanish and English.

(l) The system does not have its own automatic speech recognition module. However, it has been developed indeed a speech synthesis web service which allows storing voice files in a cache, decreasing thus response times.

(m) The state machine allows recommending similar products based on some similarity criteria (e.g. same author, category, or subject). In addition, the use of ontologies allows making connections between the different information units and offering thus recommendations based on context.

(n) This architecture is compatible with the emotional state control system, since it allows incorporating expert agents which include that feature.

After the development of SmartSeller, a comparative analysis with other existing systems was carried out. The most important aspects to take into account when designing a system of this type were considered: dialog management and memory reuse, ease of portability, database management system independence, possibility of fuzzy, temporal, or general domain queries, advising capabilities, etcetera. This analysis proved that our architecture is one of the most complete to solve the problem of the access to heterogeneous data sources. Moreover, an evaluation of the answers provided by the system determined that the 93.95% of all questions were answered by the right expert agent, and the accuracies of the Virtual Assistant and the Bookseller were equal to 86.63% and 84.69%, respectively.

In conclusion, whereas the vast majority of existing systems are focused on a very specific part of the problem of the access to data sources in natural language (e.g. only isolated queries to the database, without any chance of starting a conversation), the generic multi-agent architecture that has been proposed allows the development of conversational systems able to handle different knowledge sources simultaneously. In addition, this makes easy the extension of the systems and their adaptation to different domains. All of that, with the objective of allowing an easy transparent information access to every type of user.

6.1.4 Technology and transfer

We would like to highlight the multidisciplinary practical nature of this thesis, which has dealt with subjects related to computer science, psychology, and medicine. It has been worked on loads of areas related to artificial intelligence, natural language processing, fuzzy logic, rule based systems, ontologies, grammars, computer graphics, among others. Regarding the technology used in the development of the different systems, it has been used numerous languages such as ActionScript (Flash), C++, CSS, HTML, Java, JavaScript, OWL, SQL, or XML. In all, more than a hundred lines of code.

Finally, we would like to mention that this thesis has not only made contributions to the scientific community, but it has meant a significant technology transfer. Every chapter has resulted in the development of a commercially avail-

able system. This has served as the basis in the implementation of loads of projects, with a total investment of several hundred thousand euros, and the creation of several job positions. Among those projects we can mention Elvira [1], the virtual assistant of the University of Granada [2] (Spain) and the first virtual assistant of a Spanish university, available in Spanish and English, and *Maria*, the virtual assistant of the University of Beira Interior (Portugal), available in English and Portuguese. In addition, the results and the objectives achieved in this thesis have been published in different journals and congresses, both national and international. The full list of these publications is available in Chapter 8.

6.2 Future work

The problem of natural language processing in general, and conversational systems in particular, is an open problem of extreme relevance to the present moment, even in spite of having been tackled by loads of authors throughout several decades. As far as the content of this thesis is concerned, although good results have been achieved and a great amount of work and numerous valuable contributions have been made, there are many aspects which leave room for improvement. Next, we will mention some of those possible lines for future work:

- Regarding the emotional state of agents, we should study how the different behavior patterns of human being defined in psychology evolve throughout the time. The objective would be to perform a natural automatic adjustment of the probabilities of the rules, since until now these are kept constant along all the conversation.

- For certain projects, like the one of the Virtual Simulated Patient, it could be interesting to work on emotion filtering, since agents need not always show their emotions the way they felt in each moment. Sometimes, they could be interested in hiding them or pretending others completely different according to the social context or their personality.

[1] http://tueris.ugr.es/elvira/?lang=en
[2] http://www.ugr.es/

- Although the 3D representation model used for agents do already allows the simultaneous appearance of emotions, it would be interesting to define a series of rules that automatically controlled which emotions are incompatible with each other and therefore they should not be shown at the same time.

- With regard to closed-domain virtual assistants, one of the bottlenecks in the development of these systems is the token generation for the question recognition. Although some people have been working on making tools that make this task easy, we think that it is very important to keep working on this way, since it has direct repercussions on the reliability and quality of the final result.

- As the use of the system increases and the number of questions asked grows, the maintenance is more and more complex and all manual processes stop being effective. That is why we have developed a tool which makes the answer validation process easier. However, we must keep working on the integration of this tool with the rest of components of the framework to thus automate, as far as possible, all the error correction process.

- If we consider the answer generation part, it would be necessary to work on website monitoring for detecting changes and discovering new contents automatically.

- At this moment, the recommendations provided by the assistants are adapted to the conversation context. However, they are static, that is, they do not depend on the particular features of each user, and they do not take account of the moment (date and time) when they are offered. We think that the use of a more dynamic and collaborative approach could improve the efficiency of these recommendations.

- Regarding the access to heterogeneous data sources, it would be interesting to apply data mining algorithms in the generation of dynamic rules (e.g. to add new labels extracted automatically from the descriptions of books), or the natural language inquiry of summaries for the different attributes of the database.

- It would also be necessary to make new grammar rules to answer more types of questions (e.g. specific questions about a certain book such as "*Who wrote...?*" or "*How much is...?*").

- We could improve the Decision agent with new advising capabilities. For example, we could define new states to request the user to provide a subtopic when a category is quite big and contains many books. In addition, we could implement an algorithm that allowed classifying and sorting the search results based on diversity and representativeness.

- Although for each question we have just focused on the best interpretation found by the Bookseller, the system could be improved if the top 3 interpretations were combined and showed to the user when the best one matches only loosely the sentence. Moreover, it could be interesting to integrate the answers provided by the different expert agents.

- When it is not possible to retrieve any record from the database, the system may remove some of the restrictions specified by the user. We could improve this algorithm allowing the system to replace restrictions with alternatives based on similarity (e.g. using a similar subject).

- We could improve the recommendation algorithm used by the advising module adding a collaborative component. In this way, it would take advantage of the preferences of users with similar features.

- Finally, we could increase the capabilities of the system designing new types of expert agents. For example, for querying Wikipedia, social networks, the contact list of a mobile device, APIs that offer different types of information, and so on.

This page intentionally left blank

7

Conclusions and future work (Spanish)

En esta tesis hemos estudiado tres problemas relacionados con los sistemas conversacionales como son la mejora de su naturalidad, la sobrecarga de información y el acceso a fuentes de datos heterogéneas. A lo largo de este capítulo analizaremos qué objetivos de los planteados al comienzo de la tesis han sido alcanzados y realizaremos una serie de conclusiones. Asimismo, estableceremos algunas posibles líneas de trabajo futuro.

7.1 Conclusiones

Mediante esta tesis se ha realizado un análisis de diferentes propuestas de la literatura sobre tres problemas específicos de interés relacionados con el campo de los sistemas conversacionales: la mejora de su naturalidad, la sobrecarga de información y el acceso a fuentes de datos heterogéneas. Esto ha permitido detectar ciertas carencias existentes en dichos sistemas. En base a ellas, se han establecido una serie de requisitos imprescindibles para la resolución de cada uno de los pro-

blemas a los que nos enfrentábamos. Estos requisitos nos han permitido realizar propuestas de valor concretas, en forma de modelos genéricos, de las que hablaremos a continuación. Estos diseños han sido empleados como base en el desarrollo de sistemas específicos para la resolución de problemas reales y se encuentran disponibles comercialmente.

En general, podemos decir que se han alcanzado con éxito los tres grandes objetivos que nos marcamos al comienzo de esta tesis y se han confirmado cada una de nuestras hipótesis:

- La aplicación de sistemas basados en reglas difusas ha permitido diseñar un modelo de control de emociones para la mejora de la naturalidad de los agentes conversacionales el cual es fácil de interpretar y modificar. Los resultados obtenidos en el Apartado 3.6 del Capítulo 3 y una serie de publicaciones en revistas y congresos internacionales (Eisman y col., 2009a; López y col., 2013; Castro y col., 2007b; Váquez Granado y col., 2008; López y col., 2008; Castro y col., 2010a; Castro y col., 2010b) reflejan la calidad de nuestra propuesta.

- La organización de los contenidos del sitio web de una determinada organización mediante estructuras semánticas como las ontologías, y la aplicación de reglas y restricciones tanto a nivel de reconocimiento del lenguaje y gestión de diálogo como de generación de respuestas, han permitido la creación de un sistema que aprovecha todo el conocimiento existente y facilita el acceso inmediato y preciso a grandes cantidades de información dinámica. El análisis de resultados mostrado en el Apartado 4.5 del Capítulo 4 y varias publicaciones (Eisman y col., 2012; Eisman y col., 2009b) avalan este trabajo.

- El uso de un enfoque multiagente basado en un conjunto de agentes expertos coordinados por una serie de agentes de decisión ha permitido proporcionar una solución modular y escalable al problema del acceso a fuentes de datos heterogéneas de forma integrada, transparente, efectiva, eficiente, y atractiva. El análisis comparativo y la evaluación llevados a cabo en el Apartado 5.5 del Capítulo 5 y diversos artículos (Eisman y Castro, 2014; Eisman

y col., 2015 —*pendiente de publicación*—) ponen de manifiesto la utilidad del sistema desarrollado.

A continuación, detallaremos en qué medida y cómo se han cumplido estos objetivos generales en cada uno de los tres problemas abordados.

7.1.1 Emociones en agentes conversacionales

En relación al primero de los problemas, el de la mejora de la naturalidad de los sistemas o agentes conversacionales, se llevó a cabo un estudio sobre diferentes modelos de representación y control de emociones. Sin embargo, proyectos como el Paciente Simulado Virtual imponían una serie de requisitos que los modelos existentes no eran capaces de satisfacer. Por ello fue necesario diseñar un nuevo sistema de control que ha mejorado ciertas componentes ya incluidas en algunos modelos y ha incorporado una serie de características innovadoras:

(a) Permite reflejar ocho tipos de emociones básicas distintas en función de ocho atributos emocionales: alegría, desdén, ira, miedo, preocupación, sorpresa, tristeza y vergüenza. Estos atributos son suficientes para la mayoría de las aplicaciones, aunque el modelo puede ser extendido fácilmente para reflejar un mayor número de emociones si es necesario.

(b) Define seis tipos de personalidad diferentes en base a los ocho atributos emocionales: normal, ansiosa, depresiva, hipocondríaca, maníaca y fóbica. Estas influyen en la manera que el agente percibe su entorno. Al igual que ocurre con las emociones básicas, estas se pueden extender para cubrir las necesidades concretas de cada problema.

(c) Cada emoción básica es definida gradualmente en una escala real de 0 a 1, por lo que puede mostrarse con diferentes niveles de intensidad. Un valor 0 indica una ausencia de dicha emoción, mientras que un valor 1 indica una presencia total o máxima.

(d) El control del estado emocional del agente se lleva a cabo por medio de un conjunto de sistemas basados en reglas difusas, uno por cada uno de los

ocho atributos emocionales. Esto permite que se puedan especificar, con la ayuda de expertos en cada dominio, una serie de reglas de actualización para cada atributo. Cada regla está compuesta por un conjunto de antecedentes y un consecuente. Los antecedentes pueden estar relacionados con variables como la personalidad del agente, su estado de salud, su entorno, el tipo de pregunta realizada por el usuario, o la memoria. Por su parte, el consecuente indica el grado de variación a aplicar en el atributo emocional correspondiente cuando se dispara la regla (negativa alta/media/baja, cero, o positiva baja/media/alta). En comparación con otro tipo de aproximaciones, el uso de reglas permite comprender y justificar perfectamente el comportamiento del sistema.

(e) Mediante las reglas, es posible configurar cómo influyen ciertos factores como la personalidad o el estado emocional del agente. Por ejemplo, en el caso del Paciente Simulado Virtual, se puede controlar cómo percibe este su entorno, su estado de salud o las enfermedades que padece.

(f) Cada respuesta puede asociarse a un determinado estado emocional. Así, esta se expresará solamente cuando el agente se encuentre realmente en dicho estado. Esto evita las respuestas enlatadas que carecen de cualquier tipo de componente emocional.

(g) La estabilidad en el estado emocional del agente se consigue por medio de dos tipos de reglas. Por un lado, aquellas que no varían los atributos emocionales. Como tienen una variación nula, al dispararse disminuyen el impacto del resto de reglas. Por otra parte, las reglas que varían los atributos emocionales de acuerdo a la personalidad hacen que el agente tienda a un estado de equilibrio.

(h) La representación gráfica del agente conversacional se realizó inicialmente mediante un modelo 2D más sencillo que simplificaba un poco todo el proceso de gestión de las emociones. Posteriormente, este ha sido sustituido por un modelo 3D más avanzado que permite un mayor grado de realismo, la combinación de emociones básicas y mejora la transición entre diferentes tipos de emociones.

(i) El comportamiento no determinista del estado emocional del agente se garantiza asociando a cada regla de actualización un valor real entre 0 y 1. Este equivale a la probabilidad de tener en cuenta dicha regla en la iteración actual del sistema. Además, dichas probabilidades pueden ser actualizadas dinámicamente a medida que transcurre la conversación.

Para medir la precisión del sistema de control del estado emocional desarrollado, se realizaron una serie de entrevistas (diálogos completos) a un paciente simulado virtual. Tras ser revisadas por expertos en el dominio, estas determinaron que, en general, el comportamiento mostrado por el agente resultaba bastante natural. En el 71,8 % de los casos, el comportamiento emocional generado era el esperado, mientras que en el 28,2 % restante se detectaron algunas pequeñas variaciones en los atributos emocionales que eran opuestas a las esperadas. Aunque es cierto que fue necesario corregir los valores de probabilidad concretos asociados en un primer momento a algunas de las reglas, esto garantizaba que el sistema se comportaría de una forma no determinista.

En definitiva, los aportes realizados han permitido mejorar la naturalidad de los sistemas conversacionales y por consiguiente su credibilidad. En algunas aplicaciones, como la del Paciente Simulado Virtual, esto puede llegar a ser un aspecto fundamental puesto que el éxito en la interacción viene determinado en parte por el grado de realismo que se pueda llegar a alcanzar. En otros casos, es bastante aconsejable ya que mejora significativamente la experiencia de usuario, haciendo que todo el proceso sea mucho más natural. Por ejemplo, puede animar a un usuario a realizar una compra en un portal de comercio electrónico.

7.1.2 Asistentes virtuales de dominio cerrado

En segundo lugar, se estudió el problema de la sobrecarga de información, que suele originar cierta desorganización y dispersión en los sitios web. Se analizaron algunos de los modelos existentes para la mejora de la navegación. Estos presentaban ciertas carencias para su aplicación en un dominio como el de la Universidad de Granada. Por esta razón, se desarrollaron una metodología y un entorno de trabajo para el diseño de asistentes virtuales de dominio cerrado. Estos permitirían

acceder a gran cantidad de información de forma inmediata y precisa, incluyendo todas las características que nuestro problema requería:

(a) El diseño de diferentes tipos de unidades de información (objetos, propiedades, temas específicos, temas generales, listas de temas y preguntas guiadas) permite aprovechar el conocimiento específico que aporta el dominio. Así, es posible contestar las preguntas de una forma precisa, o algo más general, según sea necesario.

(b) Las unidades de información pueden ser organizadas jerárquicamente siguiendo la misma filosofía y principios de la programación orientada a objetos.

(c) Las respuestas pueden ser adaptadas dinámicamente en función de una serie de restricciones. De esta forma, es posible ofrecer diferente información para un mismo concepto según la fecha o la hora en que se realice la pregunta (útil para dar información sobre horarios de verano/invierno, o plazos, por ejemplo), si se ha preguntado previamente por él o es la primera vez, etcétera.

(d) El asistente funciona de forma independiente a la web y puede proporcionar información complementaria a la misma. En ocasiones, puede responder utilizando contenido resumido de la web. En otras, puede aportar información que ni siquiera esté contenida en la misma, sino en un recurso descargable o en un enlace a una web externa, por ejemplo.

(e) El conocimiento acerca de las diferentes unidades de información sobre las que puede informar el asistente se encuentra almacenado en una serie de ontologías organizadas jerárquicamente. Esta estructura permite definir ontologías independientes que facilitan la reutilización del conocimiento en diferentes dominios y son capaces de responder a preguntas de ámbito más general.

(f) El comportamiento del asistente a la hora de seleccionar una respuesta se configura mediante un sistema basado en reglas. Este permite alterar las prioridades asignadas a cada tipo de unidad de información (UI), y

por consiguiente la respuesta a proporcionar, en función de un conjunto de parámetros como la clase a la que pertenece la UI reconocida, el estado emocional actual del agente, su personalidad, o la última respuesta dada por el sistema, entre otros.

(g) El módulo de memoria y la manera en que se definen y organizan las unidades de información en la ontología permiten situar la conversación en un contexto determinado y reutilizar los objetos que hayan sido nombrados. Esto permite omitir las palabras implícitas durante el diálogo.

(h) Las unidades de información, además de estar organizadas jerárquicamente en la ontología, pueden ser interconectadas entre sí formando un grafo dirigido. Todos estos enlaces permiten realizar recomendaciones basadas en el contexto de la conversación.

(i) Para aumentar la eficiencia del sistema y que este sea capaz de responder en tiempo real a un mayor número de usuarios de forma simultánea, el conocimiento necesario para el reconocimiento de las preguntas y la generación de respuestas se encuentra precargado en memoria y su acceso se realiza mediante un conjunto de índices.

(j) El asistente puede ser configurado para comprender y responder a preguntas formuladas en cualquier idioma. Todos ellos pueden compartir la misma estructura de conocimiento para facilitar el mantenimiento. Hasta el momento, ha sido probado con éxito en tres idiomas diferentes: español, inglés y portugués.

(k) Actualmente, el asistente no cuenta con un módulo propio de reconocimiento automático del habla. Sin embargo, esta funcionalidad puede ser obtenida a través del sistema operativo desde el que el usuario utilice la aplicación. Lo que sí se ha desarrollado es un servicio web de síntesis del habla basado en un software comercial. Esto ha permitido crear una caché de archivos de sonido que disminuye los tiempos de respuesta.

(l) A través de la ontología y mediante el uso de las reglas, es posible diseñar procesos especificando las diferentes etapas o fases que los componen. De

esta forma, el asistente puede guiar a los usuarios a lo largo de esos caminos que han sido previamente definidos.

(m) El acceso restringido a la información se consigue gracias al uso de las reglas y la especificación de restricciones en las respuestas. De esta forma, es posible que el asistente deniegue al usuario el acceso a determinados recursos o le solicite que se identifique previamente. Esto no implica que el sitio web no tenga que implementar las medidas de seguridad necesarias, ya que el asistente sólo estaría redirigiendo al usuario.

(n) Aprovechando la definición de las diferentes clases de objetos existentes en la ontología, con sus propiedades o atributos específicos, es posible generar de forma automática esquemas que ofrecen de una vez toda la información relativa a un determinado objeto.

(o) Las reglas y las restricciones son perfectamente compatibles con el sistema de control del estado emocional. Así, es posible modificar las expresiones linguísticas concretas empleadas a la hora de responder a cada pregunta. Incluso, es posible modificar completamente el concepto sobre el que se va a informar.

Esta metodología y el sistema propuesto han sido utilizados como base en la creación de diferentes asistentes de dominio cerrado como los de la Universidad de Granada (España) o la Universidade da Beira Interior (Portugal). Considerando el primero de ellos, podemos decir que el asistente ha contestado a más de 500.000 preguntas de 167 países diferentes a lo largo de cinco años y medio, desde 2010. En este caso, la tasa de reconocimiento es igual al 94,39 % para las preguntas dentro de dominio (aquellas para las que ha sido diseñado el asistente), o al 76,30 % si también consideramos las preguntas fuera de dominio (sobre su vida privada, sus aficiones...), las cuales suponen un 34,41 % del total de preguntas.

En resumen, las aportaciones realizadas en este campo de los asistente virtuales de dominio cerrado han permitido desarrollar el primer asistente virtual de una universidad española, un problema que incluía ciertas características particulares: gran cantidad de información repartida por diferentes subdominios independientes; existencia de distintos colectivos con diferentes intereses (PDI —Personal

Docente e Investigador—, PAS —Personal Asociado a Servicios—, o alumnos); información dinámica que varía a lo largo del tiempo; presencia de recursos externos (normativas, resoluciones de organismos oficiales...); posibilidad de informar en varios idiomas o con diferente nivel de detalle, etcétera.

7.1.3 Acceso a fuentes de datos heterogéneas

El último de los aspectos analizados tiene que ver con el tipo de problemas que suele haber hoy en día. Lo usual es que la información se encuentre repartida por diferentes fuentes que deben ser integradas de forma simultánea y transparente para el usuario. Sin embargo, en lugar de centrarse en el concepto más amplio de interfaz de lenguaje natural que pueda ser usada por cualquier tipo de público, la mayoría de los sistemas existentes se centran en una sola parte muy específica del problema. Por ello fue necesario diseñar una nueva arquitectura genérica que ha sido utilizada para crear SmartSeller, un sistema específico para el portal de comercio electrónico de una librería. Esta arquitectura aporta numerosas ventajas en relación al resto:

(a) La integración de las diferentes fuentes de datos se realiza mediante un sistema multiagente formado por dos tipos de agentes. Existen agentes expertos independientes que están especializados en el acceso a diferentes fuentes de conocimiento. Estos son coordinados por una serie de agentes de decisión para proporcionar una respuesta final coherente al usuario.

(b) Aunque el reconocimiento mediante tokens puede ser suficiente para determinados dominios o aplicaciones, estos no ayudan a comprender exactamente la pregunta puesto que se pierde mucha informaciÃşn. Sin embargo, el uso de gramáticas semánticas, además de reducir los problemas elípticos que pueden originar ciertas consultas, nos permite sacar provecho de toda la información disponible. Así, es posible extraer todas las interpretaciones posibles de cada pregunta y determinar la mejor en base a un sistema de puntuaciones.

(c) Para mejorar la precisión del sistema, la gramática es extendida dinámicamente con una serie de reglas de producción generadas de forma automática

a partir de la información contenida en la base de datos. Además, aplicando una serie de algoritmos de poda se consigue mejorar el rendimiento de la gramática.

(d) El sistema incluye un conjunto de índices sobre la base de datos que permiten preprocesar la información para limpiarla y filtrarla, aumentando así la fiabilidad del reconocimiento y acelerando todo el proceso.

(e) Los módulos de memoria y gestión de diálogo permiten la reutilización de restricciones a lo largo de una misma conversación. De esta manera, el usuario no tiene por qué especificar todas las restricciones de golpe, sino que puede ir haciéndolo poco a poco. Qué componentes reutilizar y cómo hacerlo viene determinado por un sistema basado en reglas.

(f) El módulo de asesoramiento utiliza una máquina de estados para representar todas las posibles situaciones y etapas de la relación entre el usuario y el sistema (p.e. recibir consulta, pedir temática, pedir libro similar, recomendar novedades, recomendar libros similares...). Además, es posible asociar a cada estado una serie de acciones que deben ser llevadas a cabo al entrar o salir del mismo. Así, cuando por ejemplo el usuario no sepa muy bien qué buscar, el sistema le podrá orientar preguntándole libros que le hayan gustado.

(g) El uso de taxonomías u ontologías permiten establecer una jerarquía de categorías que enriquece la información existente en la base de datos. De esta forma, el sistema puede ir mucho más allá y responder preguntas que no podrían ser contestadas considerando solamente la información almacenada en la base de datos. Además, estas jerarquías pueden ser reutilizadas en diferentes dominios de aplicación.

(h) La gramática incluye reglas de producción que permiten reconocer conceptos difusos como "*barato*" o "*reciente*". La forma en que se interpretan estos conceptos se puede configurar para adaptarlos a cada dominio concreto.

(i) Mediante una serie de reglas de producción dinámicas, la gramática también permite reconocer e interpretar las fechas relativas incluidas en las consul-

tas temporales. En este tipo de preguntas, la información proporcionada dependerá del momento en el que se realice la consulta.

(j) El uso de una arquitectura multiagente que sigue una filosofía divide y vencerás permite la reutilización de las diferentes fuentes de conocimiento. Asimismo, las existencia de reglas de producción genéricas (p.e. las relativas a fechas o precios) y la extensión de la gramática por medio de reglas dinámicas, facilitan la adaptación del sistema a diferentes dominios.

(k) El sistema ha sido diseñado para poder incluir la información referente a diferentes idiomas aprovechando una estructura de conocimiento común. Hasta el momento, ha sido probado con éxito en español e inglés.

(l) El sistema no cuenta con un módulo propio de reconocimiento automático del habla. Sin embargo, sí se ha desarrollado un servicio web de síntesis del habla que permite almacenar en una caché los archivos de voz, disminuyendo así los tiempos de respuesta.

(m) La máquina de estados permite recomendar productos parecidos en base a ciertos criterios de similitud (p.e. mismo autor, categoría o temática). Asimismo, el uso de ontologías permite crear relaciones entre las diferentes unidades de información y ofrecer así recomendaciones basadas en el contexto.

(n) Esta arquitectura es compatible con el sistema de control del estado emocional, ya que permite incorporar agentes expertos que incluyen dicha característica.

Tras el desarrollo de SmartSeller, se realizó un análisis comparativo con otros sistemas existentes. Se consideraron los aspectos más importantes a tener en cuenta a la hora de diseñar un sistema de este tipo: gestión de diálogo y reutilización de memoria, facilidad de adaptación, independencia del sistema de gestión de la base de datos, posibilidad de consultas difusas, temporales, o de dominio general, capacidad de asesoramiento, etcétera. Este análisis demostró que nuestra arquitectura es una de las más completas para resolver el problema del acceso a fuentes de datos heterogéneas. Asimismo, una evaluación de las respuestas

proporcionadas por el sistema determinó que el 93,95 % de todas las preguntas fueron respondidas por el agente experto adecuado, y las precisiones del Asistente Virtual y el Librero eran iguales al 86,63 % y el 84,69 %, respectivamente.

Como conclusión, mientras que la gran mayoría de los sistemas existentes se centran en una parte muy específica del problema del acceso a fuentes de datos en lenguaje natural (p.e. sólo consultas aisladas a la base de datos, sin posibilidad de establecer una conversación), la arquitectura multiagente genérica que ha sido propuesta permite el desarrollo de sistemas conversacionales capaces de manejar simultáneamente diferentes fuentes de conocimiento. Además, esta facilita la extensión de los sistemas y su adaptación a diferentes dominios. Todo ello, con el objetivo de permitir un acceso a la información sencillo y transparente para cualquier tipo de usuario.

7.1.4 Tecnología y transferencia

Nos gustaría destacar el carácter práctico multidisciplinar de esta tesis en la que se han tratado temas relacionados con la informática, la psicología y la medicina. Se ha trabajado en multitud de áreas relacionadas con la inteligencia artificial, el procesamiento de lenguaje natural, la lógica difusa, los sistemas basados en reglas, las ontologías, las gramáticas, o la informática gráfica, entre otras. En cuanto a la tecnología empleada en el desarrollo de los diferentes sistemas, se han utilizado numerosos lenguajes como ActionScript (Flash), C++, CSS, HTML, Java, JavaScript, OWL, SQL, o XML. En total, más de cien mil líneas de código.

Por último, nos gustaría mencionar que esta tesis no sólo ha realizado aportaciones a la comunidad científica, sino que ha supuesto una gran transferencia tecnológica. Cada capítulo ha resultado en el desarrollo de un sistema disponible comercialmente. Esto ha servido como base para la realización de multitud de proyectos, con una inversión total de varios cientos de miles de euros, y la creación de varios puestos de trabajo. Entre esos proyectos podemos mencionar Elvira [1], el asistente virtual de la Universidad de Granada [2] (España) y primer asistente virtual de una universidad española, disponible en español e inglés, y *Maria*, el

[1] http://tueris.ugr.es/elvira/
[2] http://www.ugr.es/

asistente virtual de la Universidade da Beira Interior (Portugal), disponible en inglés y portugués. Asimismo, los resultados y los objetivos alcanzados en esta tesis han sido publicados en diferentes revistas y congresos, tanto a nivel nacional como internacional. La lista completa de dichas publicaciones se encuentra disponible en el Capítulo 8.

7.2 Trabajo futuro

El problema del procesamiento de lenguaje natural en general, y el de los sistemas conversacionales en particular, es un problema abierto y de gran actualidad, a pesar incluso de que ha sido abordado por multitud de autores a lo largo de varias décadas. En lo que respecta al contenido de esta tesis, aunque se han obtenido buenos resultados y se ha realizado una gran cantidad de trabajo y numerosas aportaciones de valor, son muchos los aspectos susceptibles de mejora o ampliación. A continuación, mencionaremos algunas de esas posibles líneas de trabajo futuro:

- En relación al estado emocional de los agentes, deberíamos estudiar cómo evolucionan a lo largo del tiempo los diferentes patrones de comportamiento del ser humano definidos en la psicología. El objetivo sería llevar a cabo un ajuste automático natural de las probabilidades de las reglas, ya que hasta el momento estas se mantienen constantes a lo largo de toda la conversación.

- Para ciertos proyectos, como el del Paciente Simulado Virtual, podría ser interesante trabajar en el filtrado de emociones, ya que los agentes no siempre tendrían por qué mostrar sus emociones tal y como las sintiesen en cada momento. En algunas situaciones, podrían estar interesados en ocultarlas o fingir otras completamente diferentes en función del contexto social o su personalidad.

- Aunque el modelo de representación 3D empleado para los agentes ya sí permite la aparición simultánea de emociones, sería interesante definir una serie de reglas que controlasen de forma automática qué emociones son

incompatibles entre sí y por tanto no deberían ser mostradas al mismo tiempo.

- Con respecto a los asistentes virtuales de dominio cerrado, uno de los cuellos de botella del desarrollo de estos sistemas es la generación de tokens para el reconocimiento de las preguntas. Aunque se ha trabajado en la contrucción de herramientas que facilitan esta labor, creemos que es muy importante seguir trabajando en esta dirección, ya que repercute directamente en la fiabilidad y calidad del resultado final.

- A medida que crece el uso del sistema y aumenta el número de preguntas realizadas, el mantenimiento se va haciendo cada vez más complejo y todos los procesos manuales dejan de ser efectivos. Por eso hemos desarrollado una herramienta que facilita el proceso de validación de respuestas. Sin embargo, debemos seguir trabajando en la integración de esta herramienta con el resto de componentes del entorno de trabajo para así automatizar, en la medida de lo posible, todo el proceso de corrección de errores.

- Si consideramos la parte de generación de las respuestas, sería necesario trabajar en la monitorización de las webs para la detección de cambios y el descubrimiento de nuevos contenidos de forma automática.

- Actualmente, las recomendaciones proporcionadas por los asistentes están adaptadas al contexto de la conversación. Sin embargo, son estáticas, es decir, no dependen de las características concretas de cada usuario, y no tienen en cuenta el momento (fecha y hora) en el que son ofrecidas. Creemos que la utilización de un enfoque más dinámico y colaborativo podría mejorar la eficacia de estas recomendaciones.

- En cuanto al acceso a fuentes de datos heterogéneas, sería interesante aplicar algoritmos de minería de datos en la generación de reglas dinámicas (p.e. para añadir nuevas etiquetas extraídas automáticamente de las descripciones de los libros), o la consulta en lenguaje natural de resúmenes para los diferentes atributos de la base de datos.

- También sería necesario crear nuevas reglas gramaticales para responder a más tipos de preguntas (p.e. preguntas específicas sobre un determinado libro tales como "*¿Quién escribió...?*" o "*¿Cuánto cuesta...?*"...).

- Podríamos mejorar el agente de Decisión con nuevas capacidades de asesoramiento. Por ejemplo, podríamos definir nuevos estados para solicitar al usuario que proporcione una subtemática cuando una categoría es bastante grande y contiene muchos libros. Además, podríamos implementar un algoritmo que permitiese clasificar y ordenar los resultados de búsqueda en base a la diversidad y representatividad.

- Aunque para cada pregunta nos hemos centrado solamente en la mejor interpretación encontrada por el Librero, el sistema podría mejorarse si se combinasen y se mostrasen al usuario las tres mejores interpretaciones cuando la mejor casa sólo vagamente con la frase. Además, podría ser interesante integrar las respuestas proporcionadas por los diferentes agentes expertos.

- Cuando no es posible recuperar ningún registro de la base de datos, el sistema puede eliminar algunas de las restricciones especificadas por el usuario. Podríamos mejorar este algoritmo permitiendo al sistema reemplazar las restricciones por alternativas en base a la similitud (p.e. utilizando una temática similar).

- Podríamos mejorar el algoritmo de recomendación utilizado por el módulo de asesoramiento añadiendo una componente colaborativa. De esta manera, le sacaría partido a las preferencias de los usuarios con características similares.

- Finalmente, podríamos aumentar las capacidades del sistema diseñando nuevos tipos de agentes expertos. Por ejemplo, para la consulta de la Wikipedia, las redes sociales, la agenda de contactos de un dispositivo móvil, APIs que ofrezcan diversos tipos de información, etcétera.

This page intentionally left blank

8

Publications

All the work included in this memory has been published in different journals and congresses, both national and international:

Castro et al., 2007a. Juan L. Castro, **Eduardo M. Eisman**, Víctor López, María I. Navarro, and Jose M. Zurita (2007a). Paciente Simulado Virtual. *'Fin de Semana de las TIC' (FISETIC 2007)*. Almuñécar, Granada, Spain.

Castro et al., 2007b. Juan L. Castro, **Eduardo M. Eisman**, Víctor López, María I. Navarro, and Jose M. Zurita (2007b). Virtual Simulated Patient: an alternative approach to role-based training in clinical interviews. *Proceedings of the 13th Annual Meeting of the Society in Europe for Simulation Applied to Medicine (SESAM 2007)*. Copenhague, Denmark.

Váquez Granado et al., 2008. Javier Váquez Granado, Rocío Ruiz, Ana Gonzáles, Carlos González, Javier Guerrero, Juan Luis Castro, Jose Manuel Zurita, Víctor López, **Eduardo Eisman**, and María Isabel Navarro (2008). Virtual Simulated Patient: A Novel Health Professionals' Training. *Proceedings of the 14th Annual Meeting of the Society in Europe for Simulation Applied to Medicine (SESAM 2008)*. Hatfield, The United Kingdom.

Eisman et al., 2008. **Eduardo M. Eisman**, Víctor López, and Juan Luis Castro (2008). A Natural Language Virtual Tutoring System. *Proceedings of the IADIS International Conference on e-Learning (IADIS 2008)*. Amsterdam, The Netherlands, pp. 99–103. URL: http://www.researchgate.net/profile/Juan_Castro15/publication/220969912_A_Natural_Language_Virtual_Tutoring_System/links/02e7e52b38df7400d1000000.pdf.

López et al., 2008. V. López, **E. M. Eisman**, and J. L. Castro (2008). A Tool for Training Primary Health Care Medical Students: The Virtual Simulated Patient. *ICTAI '08: Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence*. Vol. 2. Washington, DC, USA: IEEE Computer Society, pp. 194–201. isbn: 978-0-7695-3440-4. DOI: 10.1109/ICTAI.2008.50.

Eisman et al., 2009b. **Eduardo M. Eisman**, Víctor López, and Juan Luis Castro (2009b). Intelligent Web Navigation Using Virtual Assistants. *Proceedings of the 7th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems. The Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*. Pasadena, California, USA. URL: http://www.cs.tu-dortmund.de/nps/de/Forschung/Publikationen/Graue_Reihe1/Ver___ffentlichungen_2009/825.pdf#page=87.

Eisman et al., 2009a. **Eduardo M. Eisman**, Víctor López, and Juan Luis Castro (2009a). Controlling the emotional state of an embodied conversational agent with a dynamic probabilistic fuzzy rules based system. *Expert Systems with Applications* 36.6, pp. 9698 –9708. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2009.02.015.

- Journal Impact Factor (2009 JCR Science Edition): 2.908.
- Journal Ranking (2009 JCR Science Edition):
  - Computer Science, Artificial Intelligence: 15 / 103 (Q1).
  - Engineering, Electrical & Electronic: 16 / 246 (Q1).
  - Operations Research & Management Science: 3 / 73 (Q1).

Castro et al., 2010a. J. L. Castro, **E. M. Eisman**, V. López, M. Navarro, and J. M. Zurita (2010a). A multilingual virtual simulated patient to improve health professionals 'training'. *Proceedings of the 16th Annual Meeting of the Society in Europe for Simulation Applied to Medicine (SESAM 2010)*. Groningen, The Netherlands.

Eisman et al., 2010. **Eduardo M. Eisman**, María I. Navarro, Víctor López, Juan L. Castro, and Jose M. Zurita (2010). Multilingual Virtual Simulated Patient. *Proceedings of the 7th IAVANTE International Symposium on Medical Simulation*. CMAT (Complejo Multifuncional Avanzado de Simulación e Innovación Tecnológica). Granada, Spain: Fundación IAVANTE.

Castro et al., 2010b. Juan L. Castro, María I. Navarro, Víctor López, **Eduardo M. Eisman**, and José M. Zurita (2010b). A multilingual virtual simulated patient framework for training primary health care students. *Proceedings of the International Conference on Health and Medical Informatics (ICHMI 2010)*. Vol. 4. 8. Paris, France: World Academy of Science, Engineering and Technology. URL: http://waset.org/publications/2580/a-multilingual-virtual-simulated-patient-framework-for-training-primary-health-care-students.

Eisman et al., 2012. **Eduardo M. Eisman**, Víctor López, and Juan Luis Castro (2012). A framework for designing closed domain virtual assistants. *Expert Systems with Applications* 39.3, pp. 3135–3144. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2011.08.177.

- Journal Impact Factor (2012 JCR Science Edition): 1.854.
- Journal Ranking (2012 JCR Science Edition):
  - Computer Science, Artificial Intelligence: 31 / 115 (Q2).
  - Engineering, Electrical & Electronic: 56 / 243 (Q1).
  - Operations Research & Management Science: 13 / 79 (Q1).

López Salazar et al., 2012. Víctor López Salazar, **Eduardo M. Eisman Cabeza**, Juan Luis Castro Peña, and Jose Manuel Zurita López (2012). A case based reasoning model for multilingual language generation in dialogues.

Expert Systems with Applications 39.8, pp. 7330 –7337. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.01.085.

- Journal Impact Factor (2012 JCR Science Edition): 1.854.
- Journal Ranking (2012 JCR Science Edition):
  - Computer Science, Artificial Intelligence: 31 / 115 (Q2).
  - Engineering, Electrical & Electronic: 56 / 243 (Q1).
  - Operations Research & Management Science: 13 / 79 (Q1).

López et al., 2013. Víctor López, **Eduardo M. Eisman**, María Navarro, Jose Manuel Zurita, and Juan Luis Castro (2013). Un Paciente Simulado Virtual Multilingüe para la formación en medicina. *IE Comunicaciones: Revista Iberoamericana de Informática Educativa* 18, pp. 29–40. ISSN: 1699-4574. URL: http://dialnet.unirioja.es/servlet/articulo?codigo=4468676.

Medina et al., 2013. Javier Medina, **Eduardo M. Eisman**, and Juan Luis Castro (2013). Asistentes virtuales en plataformas 3.0. *IE Comunicaciones: Revista Iberoamericana de Informática Educativa* 18, pp. 41–49. ISSN: 1699-4574. URL: http://dialnet.unirioja.es/servlet/articulo?codigo=4468692.

Eisman and Castro, 2014. **Eduardo M. Eisman**, and Juan Luis Castro (2014). Sistema conversacional multiagente para acceder a fuentes heterogéneas. *Aplicaciones Multidisciplinares de Sistemas de Diálogo.* ISBN: 978-84-940128-5-3.

Eisman et al., 2015. **Eduardo M. Eisman**, María Navarro, and Juan Luis Castro (2015). A multi-agent conversational system with heterogeneous data sources access. *Submitted (under second review).*

Additionally, I have collaborated in the publication of other works related to data mining, information retrieval systems and their applications:

Balderas et al., 2005. Marco-Antonio Balderas, Fernando Berzal, Juan-Carlos Cubero, **Eduardo Eisman**, and Nicolás Marín (2005). Discovering hidden association rules. *KDD-2005 Workshop on Data Mining*

Methods for Anomaly Detection, Detection. Chicago, pp. 13–20. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.8718&rep=rep1&type=pdf#page=15.

Moreo et al., 2013. A. Moreo, **E.M. Eisman**, J.L. Castro, and J.M. Zurita (2013). Learning regular expressions to template-based FAQ retrieval systems. *Knowledge-Based Systems* 53.0, pp. 108–128. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2013.08.018.

- Journal Impact Factor (2013 JCR Science Edition): 3.058.
- Journal Ranking (2013 JCR Science Edition):
  - Computer Science, Artificial Intelligence: 15 / 121 (Q1).

Although the contents of the latter papers have not been explicitly included in this memory, they have contributed to my training in those areas of artificial intelligence.

This page intentionally left blank

References

Aamodt, Agnar and Enric Plaza (1994). Case-based reasoning; Foundational issues, methodological variations, and system approaches. *AI COMMUNICATIONS* 7.1, pp. 39–59. URL: http://www.idi.ntnu.no/emner/tdt4173/papers/Aamodt_1994_Case.pdf (page 137).

Abbattista, Fabio, Graziano Catucci, Giovanni Semeraro, and Fabio Zambetta (2004). SAMIR: A Smart 3D Assistant on the Web. *PsychNology Journal* 2.1, pp. 43–60. URL: http://www.psychnology.org/File/PSYCHNOLOGY_JOURNAL_2_1_ABBATTISTA.pdf (pages 9, 32, 89, 138).

Adolphs, Peter, Anton Benz, Núria Bertomeu Castelló, Xiwen Cheng, Tina Klüwer, Manfred Krifka, Alexandra Strekalova, Hans Uszkoreit, and Feiyu Xu (2011). Conversational Agents in a Virtual World. *KI 2011: Advances in Artificial Intelligence*. Ed. by Joscha Bach and Stefan Edelkamp. Vol. 7006. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 38–49. ISBN: 978-3-642-24454-4. DOI: 10.1007/978-3-642-24455-1_4 (page 181).

Ahad, Abdul, Bernhard Jung, and Helmut Prendinger (2007). Neva: A Conversational Agent Based Interface for Library Information Systems. *Proceedings of the Second IASTED International Conference on Human Computer Interaction*. IASTED-HCI '07. Chamonix, France: ACTA Press, pp. 124–129. ISBN: 978-0-88986-655-3. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.1664&rep=rep1&type=pdf (page 140).

Allison, DeeAnn (2012). Chatbots in the library: is it time? *Library Hi Tech* 30.1, pp. 95–107. DOI: 10.1108/07378831211213238 (page 144).

Amble, Tore (2000). BusTUC: a natural language bus route oracle. *Proceedings of the sixth conference on Applied natural language processing*. ANLC '00. Seattle, Washington: Association for Computational Linguistics, pp. 1–6. DOI: 10.3115/974147.974148 (pages 134, 136).

Androutsopoulos, I., G. Ritchie, and P. Thanisch (1993). MASQUE/SQL: an efficient and portable natural language query interface for relational databases. *Proceedings of the 6th international conference on Industrial and engineering applications of artificial intelligence and expert systems*. IEA/AIE'93. Edinburgh, Scotland: Gordon & Breach Science Publishers, pp. 327–330. ISBN: 2-88124-604-4. URL: http://homepages.inf.ed.ac.uk/rbf/MY_DAI_OLD_FTP/rp639.pdf (pages 134, 135, 149).

Androutsopoulos, I., G.D. Ritchie, and P. Thanisch (1995). Natural Language Interfaces to Databases - An Introduction. *Natural Language Engineering* 1 (01), pp. 29–81. ISSN: 1469-8110. DOI: 10.1017/S135132490000005X (pages 11, 33, 145, 146).

Armstrong, Robert, Dayne Freitag, Thorsten Joachims, and Tom Mitchell (1995). WebWatcher: A Learning Apprentice for the World Wide Web. *AAAI Spring Symposium on Information Gathering*, pp. 6–12. URL: http://www.aaai.org/Papers/Symposia/Spring/1995/SS-95-08/SS95-08-002.pdf (pages 8, 31, 86).

Artificial Intelligence Corporation (1982). Query Systems for End Users. *EDP Analyzer* 20 (9) (pages 131, 133).

Auer, Sören, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives (2007). DBpedia: A Nucleus for a Web of Open Data. *The Semantic Web*. Vol. 4825. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 722–735. ISBN: 978-3-540-76297-3. DOI: 10.1007/978-3-540-76298-0_52 (pages 143, 198, 209).

Auxerre, P. and R. Inder (1986). MASQUE Modular Answering System for Queries in English-User's Manual. Tech. rep. Technical Report AIAI/SR/10, Artificial Intelligence Applications Institute, University of Edinburgh (page 135).

Balderas, Marco-Antonio, Fernando Berzal, Juan-Carlos Cubero, Eduardo Eisman, and Nicolás Marín (2005). Discovering hidden association rules. *KDD-2005 Workshop on Data Mining Methods for Anomaly Detection*. Chicago, pp. 13–20. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.8718&rep=rep1&type=pdf#page=15 (page 250).

Ball, Gene, Dan Ling, David Kurlander, John Miller, David Pugh, Tim Skelly, Andy Stankosky, David Thiel, Maarten Van Dantzich, and Trace Wax (1997). Lifelike Computer Characters: the Persona project at Microsoft Research. Cambridge, MA, USA: MIT Press, pp. 191–222. ISBN: 0-262-52234-9. URL: http://www.kurlander.net/DJ/Pubs/peedycha.pdf (pages 8, 31, 87).

Ballard, Bruce W. and Douglas E. Stumberger (1986). Semantic acquisition in TELI: a transportable, user-customized natural language processor. *Meeting of the Association for Computational Linguistics*. URL: http://dl.acm.org/citation.cfm?id=981136 (pages 134, 149).

Ballard, Bruce W., John C. Lusth, and Nancy L. Tinkham (1984). LDC-1: a transportable, knowledge-based natural language processor for office environments. *ACM Transactions on Information Systems* 2 (1), pp. 1–25. DOI: 10.1145/357417.357418 (page 134).

Bartneck, Christoph (2002). Integrating the OCC Model of Emotions in Embodied Characters. *Proceedings of the Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.4323&rep=rep1&type=pdf (pages 7, 29, 51).

Benz, Anton, Núria Bertomeu, and Alexandra Strekalova (2011). A decision-theoretic approach to finding optimal responses to over-constrained queries in a conceptual search space. *Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue*. Los Angeles, California, pp. 37–46. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.431.3399&rep=rep1&type=pdf#page=37 (page 181).

Bernstein, Abraham and Esther Kaufmann (2006). GINO - A Guided Input Natural Language Ontology Editor. *The Semantic Web - ISWC 2006*. Vol. 4273.

Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 144–157. ISBN: 978-3-540-49029-6. DOI: 10.1007/11926078_11 (page 139).

Bernstein, Abraham, Esther Kaufmann, and Christian Kaiser (2005). Querying the Semantic Web with Ginseng: A Guided Input Natural Language Search Engine. *15th Workshop on Information Technologies and Systems, Las Vegas, NV*, pp. 112–126. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.8562&rep=rep1&type=pdf (page 138).

Bertomeu, Núria and Anton Benz (2009). Ontology-Based Information States for an Artificial Sales Agent. *Proceedings of DiaHolmia. 2009 Workshop on the Semantics and Pragmatics of Dialogue*. Stockholm, Sweden, pp. 135–136. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.8892&rep=rep1&type=pdf#page=147 (page 181).

Bertomeu, Núria, Hans Uszkoreit, Anette Frank, Hans-Ulrich Krieger, and Brigitte Jörg (2006). Contextual phenomena and thematic relations in database QA dialogues: results from a Wizard-of-Oz Experiment. *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*. IQA '06. New York City, NY: Association for Computational Linguistics, pp. 1–8. URL: http://dl.acm.org/citation.cfm?id=1641579.1641580 (page 181).

Bertomeu Castelló, Núria (2012). Finding Optimal Presentation Sequences for a Conversational Recommender System. *Advances in Computational Intelligence*. Ed. by Salvatore Greco, Bernadette Bouchon-Meunier, Giulianella Coletti, Mario Fedrizzi, Benedetto Matarazzo, and Ronald R. Yager. Vol. 300. Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 328–337. ISBN: 978-3-642-31723-1. DOI: 10.1007/978-3-642-31724-8_34 (page 181).

Bertomeu Castelló, Núria and Anton Benz (2009). Annotation of joint projects and information states in human-npc dialogues. *A survey of corpus-based research*, pp. 723–740. ISBN: 978-84-692-2198-3. URL: http://www.um.es/lacell/aelinco/contenido/pdf/49.pdf (page 181).

Bizer, Christian, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann (2009). DBpedia - A crystalliza-

tion point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web* 7.3. The Web of Data, pp. 154 –165. ISSN: 1570-8268. DOI: 10.1016/j.websem.2009.07.002 (page 143).

Bohus, Dan and Alexander I. Rudnicky (2003). RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. URL: http://repository.cmu.edu/cgi/viewcontent.cgi?article=2391&context=compsci (page 140).

Briscoe, Ted, John Carroll, and Rebecca Watson (2006). The Second Release of the RASP System. *Proceedings of the COLING/ACL on Interactive Presentation Sessions*. COLING-ACL '06. Sydney, Australia: Association for Computational Linguistics, pp. 77–80. DOI: 10.3115/1225403.1225423 (page 143).

Bronnenberg, W., H. Bunt, J. Landsbergen, R. Scha, W. Schoenmakers, and E. van Utteren (1980). The question answering system PHLIQA1. *In L. Bolc (ed.) Natural language question answering systems* 2.29, pp. 217–305 (pages 133, 148).

Budzikowska, Margo, Joyce Chai, Sunil Govindappa, Veronika Horvath, Nanda Kambhatla, Nicolas Nicolov, and Wlodek Zadrozny (2001). Conversational Sales Assistant for Online Shopping. *Proceedings of the First International Conference on Human Language Technology Research*. HLT '01. San Diego: Association for Computational Linguistics, pp. 1–2. DOI: 10.3115/1072133.1072146 (pages 136, 148).

Cassell, Justine, Joseph Sullivan, S. Prevost, and Elizabeth Churchill (2000). Embodied conversational agents. Cambridge, MA, USA: MIT Press. ISBN: 9780262032780 (pages 9, 32, 89).

Castro, J. L., E. M. Eisman, V. López, M. Navarro, and J. M. Zurita (2010a). A multilingual virtual simulated patient to improve health professionals 'training'. *Proceedings of the 16th Annual Meeting of the Society in Europe for Simulation Applied to Medicine (SESAM 2010)*. Groningen, The Netherlands (pages 214, 230, 249).

Castro, Juan L., Eduardo M. Eisman, Víctor López, María I. Navarro, and Jose M. Zurita (2007a). Paciente Simulado Virtual. *'Fin de Semana de las TIC' (FISETIC 2007)*. Almuñécar, Granada, Spain (page 247).

Castro, Juan L., Eduardo M. Eisman, Víctor López, María I. Navarro, and Jose M. Zurita (2007b). Virtual Simulated Patient: an alternative approach to role-based training in clinical interviews. *Proceedings of the 13th Annual Meeting of the Society in Europe for Simulation Applied to Medicine (SESAM 2007)*. Copenhague, Denmark (pages 214, 230, 247).

Castro, Juan L., María I. Navarro, Víctor López, Eduardo M. Eisman, and José M. Zurita (2010b). A multilingual virtual simulated patient framework for training primary health care students. *Proceedings of the International Conference on Health and Medical Informatics (ICHMI 2010)*. Vol. 4. 8. Paris, France: World Academy of Science, Engineering and Technology. URL: http://waset.org/publications/2580/a-multilingual-virtual-simulated-patient-framework-for-training-primary-health-care-students (pages 46, 214, 230, 249).

Chai, Joyce, Veronika Horvath, Nanda Kambhatla, Nicolas Nicolov, and Margo Stys-Budzikowska (2001a). A Conversational Interface for Online Shopping. *Proceedings of the First International Conference on Human Language Technology Research*. HLT '01. San Diego: Association for Computational Linguistics, pp. 1–4. DOI: 10.3115/1072133.1072145 (pages 137, 148).

Chai, Joyce, Jimmy Lin, Wlodek Zadrozny, Yiming Ye, Margo Stys-Budzikowska, Veronika Horvath, Nanda Kambhatla, and Catherine Wolf (2001b). The Role of a Natural Language Conversational Interface in Online Sales: A Case Study. *International Journal of Speech Technology* 4.3-4, pp. 285–295. ISSN: 1381-2416. DOI: 10.1023/A:1011316909641 (pages 2, 24, 128, 137, 148).

Chai, Joyce, Veronika Horvath, Nicolas Nicolov, Margo Stys, Nanda Kambhatla, Wlodek Zadrozny, and Prem Melville (2002). Natural Language Assistant: A Dialog System for Online Product Recommendation. *AI Magazine* 23.2, p. 63. DOI: 10.1609/aimag.v23i2.1641 (pages 12, 13, 35, 36, 137, 148, 189).

Chai, Joyce Yue, Malgorzata Budzikowska, Veronika Horvath, Nicolas Nicolov, Nanda Kambhatla, and Wlodek Zadrozny (2001c). Natural Language Sales Assistant - A Web-based Dialog System for Online Sales. *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference*. AAAI Press, pp. 19–26. ISBN: 1-57735-134-7. URL: http://www.aaai.org/Papers/IAAI/2001/IAAI01-003.pdf (pages 2, 24, 128, 137, 148).

Cimiano, Philipp, Peter Haase, Jörg Heizmann, and Matthias Mantel (2007a). ORAKEL: A Portable Natural Language Interface to Knowledge Bases. Tech. rep. URL: http://pub.uni-bielefeld.de/download/2497310/2525113 (pages 141, 149).

Cimiano, Philipp, Peter Haase, and Jörg Heizmann (2007b). Porting Natural Language Interfaces between Domains –An Experimental User Study with the ORAKEL System–. *Proceedings of the 12th International Conference on Intelligent User Interfaces*. IUI '07. Honolulu, Hawaii, USA: ACM, pp. 180–189. ISBN: 1-59593-481-2. DOI: 10.1145/1216295.1216330 (pages 141, 149).

Cimiano, Philipp, Peter Haase, Jörg Heizmann, Matthias Mantel, and Rudi Studer (2008). Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system. *Data & Knowledge Engineering* 65.2. Including Special Section: 3rd XML Schema and Data Management Workshop (XSDM 2006) - Five selected and extended papers, pp. 325 –354. ISSN: 0169-023X. DOI: 10.1016/j.datak.2007.10.007 (pages 13, 36, 141, 149, 189).

Cockburn, Andy, Saul Greenberg, Bruce McKenzie, Michael Jasonsmith, and Shaun Kaasten (1999). WebView: A Graphical Aid for Revisiting Web Pages. *IZCHI'99: Australian Conference on Human-Computer Interaction*. Wagga Wagga, Australia. URL: http://grouplab.cpsc.ucalgary.ca/grouplab/uploads/Publications/Publications/1999-WebView.Ozchi.pdf (pages 8, 31, 89).

Codd, E. F. (1974). Seven Steps to Rendezvous with the Casual User. *IFIP Working Conference Data Base Management*, pp. 179–200 (pages 132, 149).

Cullinane Corporation (1980). IQS Summary Description. *Computational Linguistics* (page 133).

Damerau, Fred J. (1981). Operating Statistics for The Transformational Question Answering System. *Computational Linguistics* 7 (1), pp. 30–42. URL: http://dl.acm.org/citation.cfm?id=972884 (page 133).

Damljanovic, Danica, Milan Agatonovic, and Hamish Cunningham (2010). Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-Based Lookup through the User Interaction. *The Semantic Web: Research and Applications*. Ed. by Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache. Vol. 6088. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 106–120. ISBN: 978-3-642-13485-2. DOI: 10.1007/978-3-642-13486-9_8 (page 142).

Damljanovic, Danica, Milan Agatonovic, and Hamish Cunningham (2012). FREyA: An Interactive Way of Querying Linked Data Using Natural Language. *The Semantic Web: ESWC 2011 Workshops*. Vol. 7117. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 125–138. ISBN: 978-3-642-25952-4. DOI: 10.1007/978-3-642-25953-1_11 (pages 13, 35, 142, 147).

Dittenbach, Michael, Dieter Merkl, and Helmut Berger (2003). A Natural Language Query Interface for Tourism Information. *Information and Communication Technologies in Tourism 2003*, pp. 152–162. ISBN: 978-3-211-83910-0. URL: http://www.researchgate.net/profile/Dieter_Merkl/publication/228787128_A_natural_language_query_interface_for_tourism_information/links/0deec529711f75e479000000.pdf (page 137).

Egges, Arjan, Sumedha Kshirsagar, and Nadia Magnenat-Thalmann (2003). A Model for Personality and Emotion Simulation. *Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems 2003. Oxford, UK, September 3-5*. Vol. 2773, PART 1. Conference Code: 63855, pp. 453–461. ISBN: 03029743 (ISSN). DOI: 10.1007/978-3-540-45224-9_63 (pages 8, 30, 54).

Eisman, Eduardo M. and Juan Luis Castro (2014). Sistema conversacional multiagente para acceder a fuentes heterogéneas. *Aplicaciones Multidisciplinares de Sistemas de Diálogo*. ISBN: 978-84-940128-5-3 (pages 214, 230, 250).

Eisman, Eduardo M., Víctor López, and Juan Luis Castro (2008). A Natural Language Virtual Tutoring System. *Proceedings of the IADIS International Conference on e-Learning (IADIS 2008)*. Amsterdam, The Netherlands, pp. 99–103. URL: http://www.researchgate.net/profile/Juan_Castro15/publication/220969912_A_Natural_Language_Virtual_Tutoring_System/links/02e7e52b38df7400d1000000.pdf (page 248).

Eisman, Eduardo M., Víctor López, and Juan Luis Castro (2009a). Controlling the emotional state of an embodied conversational agent with a dynamic probabilistic fuzzy rules based system. *Expert Systems with Applications* 36.6, pp. 9698 –9708. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2009.02.015 (pages 107, 153, 157, 214, 230, 248).

Eisman, Eduardo M., Víctor López, and Juan Luis Castro (2009b). Intelligent Web Navigation Using Virtual Assistants. *Proceedings of the 7th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems. The Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*. Pasadena, California, USA. URL: http://www.cs.tu-dortmund.de/nps/de/Forschung/Publikationen/Graue_Reihe1/Ver__ffentlichungen_2009/825.pdf#page=87 (pages 214, 230, 248).

Eisman, Eduardo M., María I. Navarro, Víctor López, Juan L. Castro, and Jose M. Zurita (2010). Multilingual Virtual Simulated Patient. *Proceedings of the 7th IAVANTE International Symposium on Medical Simulation*. CMAT (Complejo Multifuncional Avanzado de Simulación e Innovación Tecnológica). Granada, Spain: Fundación IAVANTE (page 249).

Eisman, Eduardo M., Víctor López, and Juan Luis Castro (2012). A framework for designing closed domain virtual assistants. *Expert Systems with Applications* 39.3, pp. 3135 –3144. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2011.08.177 (pages 150, 214, 230, 249).

Ekman, P. (1984). Expression and the nature of emotion. *Approaches to Emotion*, pp. 319–343 (pages 7, 29, 48, 56).

El-Nasr, Magy Seif, John Yen, and Thomas R. Ioerger (2000). FLAME - Fuzzy Logic Adaptive Model of Emotions. *Autonomous Agents and Multi-Agent Systems* 3.3, pp. 219–257. DOI: 10.1023/A:1010030809960 (pages 7, 30, 51, 53).

Elliott, Clark Davidson (1992). The affective reasoner: a process model of emotions in a multi-agent system. PhD thesis. Evanston, Illinois, USA: Northwestern University. URL: http://dl.acm.org/citation.cfm?id=142741 (pages 7, 30, 52).

Fellbaum, Christiane (1998). WordNet. Wiley Online Library. DOI: 10.1002/9781405198431.wbeal1285 (pages 139, 141, 143, 195).

Gill, Arthur et al. (1962). Introduction to the theory of finite-state machines (page 178).

Giordani, Alessandra (2008). Mapping Natural Language into SQL in a NLIDB. *Natural Language and Information Systems*. Vol. 5039. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 367–371. ISBN: 978-3-540-69857-9. DOI: 10.1007/978-3-540-69858-6_46 (page 142).

Giordani, Alessandra and Alessandro Moschitti (2010). Semantic Mapping between Natural Language Questions and SQL Queries via Syntactic Pairing. *Natural Language Processing and Information Systems*. Vol. 5723. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 207–221. ISBN: 978-3-642-12549-2. DOI: 10.1007/978-3-642-12550-8_17 (page 143).

Glass, J., J. Chang, and M. McCandless (1996). A probabilistic framework for feature-based speech recognition. *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. Vol. 4, 2277–2280 vol.4. DOI: 10.1109/ICSLP.1996.607261 (page 191).

Goddeau, David, Eric Brill, James R. Glass, Christine Pao, Michael Phillips, Joseph Polifroni, Stephanie Seneff, and Victor W. Zue (1994). Galaxy: A human-language interface to on-line travel information. *Third International Conference on Spoken Language Processing* (page 135).

González B., J. Javier, Rodolfo A. Pazos R., Alexander Gelbukh, Grigori Sidorov, Hector Fraire H., and I.Cristina Cruz C. (2007). Prepositions and Conjunc-

tions in a Natural Language Interfaces to Databases. *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*. Vol. 4743. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 173–182. ISBN: 978-3-540-74766-6. DOI: 10.1007/978-3-540-74767-3_19 (page 139).

Grosz, Barbara J., Douglas E. Appelt, Paul A. Martin, and Fernando C.N. Pereira (1987). TEAM: An experiment in the design of transportable natural-language interfaces. *Artificial Intelligence* 32.2, pp. 173 –243. ISSN: 0004-3702. DOI: 10.1016/0004-3702(87)90011-7 (pages 134, 149).

Gruber, Thomas R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition* 5.2, pp. 199 –220. ISSN: 1042-8143. DOI: 10.1006/knac.1993.1008 (pages 95, 154).

Hafner, Carole D. (1984). Interaction of Knowledge Sources in a Portable Natural Language Interface. *International Conference on Computational Linguistics*, pp. 57–60. DOI: 10.3115/980491.980505 (pages 134, 148).

Hallett, Catalina (2006). Generic Querying of Relational Databases using Natural Language Generation Techniques. *Proceedings of the Fourth International Natural Language Generation Conference*. INLG '06. Sydney, Australia: Association for Computational Linguistics, pp. 95–102. ISBN: 1-932432-72-8. URL: http://dl.acm.org/citation.cfm?id=1706289 (page 140).

Harris, Larry R. (1978). The ROBOT System: Natural Language Processing Applied to Data Base Query. *Proceedings of the 1978 Annual Conference*. ACM '78. Washington, D.C., USA: ACM, pp. 165–172. ISBN: 0-89791-000-1. DOI: 10.1145/800127.804087 (page 133).

Hendrix, Gary G., Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum (1978). Developing a natural language interface to complex data. *ACM Trans. Database Syst.* 3.2, pp. 105–147. ISSN: 0362-5915. DOI: 10.1145/320251.320253 (pages 11, 34, 133, 148).

Hinrichs, Erhard W. (1988). Tense, Quantifiers, and Contexts. *Comput. Linguist.* 14.2, pp. 3–14. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=55057 (pages 135, 149).

Huang, Bei-Bei, Guigang Zhang, and P.C.-Y. Sheu (2008). A Natural Language Database Interface Based on a Probabilistic Context Free Grammar. *Semantic Computing and Systems, 2008. WSCS '08. IEEE International Workshop on*, pp. 155–162. DOI: 10.1109/WSCS.2008.14 (page 142).

Huberman, Bernardo A., Peter L. T. Pirolli, James E. Pitkow, and Rajan M. Lukose (1998). Strong Regularities in World Wide Web Surfing. *Science* 280.5360, pp. 95–97. DOI: 10.1126/science.280.5360.95 (pages 2, 24, 128).

Jaczynski, Michel and Brigitte Trousse (1998). WWW assisted browsing by reusing past navigations of a group of users. *EWCBR '98: Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning*. Vol. 1488. Lecture Notes in Computer Science. London, UK: Springer, pp. 160–171. ISBN: 3-540-64990-5. DOI: 10.1007/BFb0056330 (pages 9, 31, 88).

Jurafsky, Daniel and James H. Martin (2000). Speech and Language Processing. 2nd Edition. ISBN: 978-0131873216.

Jusoh, S. and H.M. Al-Fawareh (2007). Natural language interface for online sales systems. *Intelligent and Advanced Systems, 2007. ICIAS 2007. International Conference on*, pp. 224–228. DOI: 10.1109/ICIAS.2007.4658379 (page 140).

Kaufmann, Esther and Abraham Bernstein (2010). Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web* 8.4. Semantic Web Challenge 2009 User Interaction in Semantic Web research, pp. 377 –393. ISSN: 1570-8268. DOI: 10.1016/j.websem.2010.06.001 (pages 2, 24, 128).

Kaufmann, Esther, Abraham Bernstein, and Renato Zumstein (2006). Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs. *In: 5th ISWC*. Springer, pp. 980–981. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.6633&rep=rep1&type=pdf (page 140).

Kaufmann, Esther, Abraham Bernstein, and Lorenz Fischer (2007). NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies. ESWC Zurich. URL: https://files.ifi.uzh.ch/ddis/oldweb/

ddis/staff/goehring/btw/files/Kaufmann_NLP-Reduce_ESWC2007.pdf (page 140).

Kautz, Henry and Bart Selman (1998). BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. *AIPS98 Workshop on Planning as Combinatorial Search*, pp. 58–60. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.1189&rep=rep1&type=pdf (page 138).

Kellogg, Charles, John Burger, Timothy Diller, and Kenneth Fogt (1971). The Converse Natural Language Data Management System: Current Status and Plans. *Proceedings of the 1971 International ACM SIGIR Conference on Information Storage and Retrieval*. SIGIR '71. College Park, Maryland: ACM, pp. 33–46. DOI: 10.1145/511285.511290 (page 132).

Khamis, R. and S. Shatnawi (2010). Toward enhanced Natural Language Processing to databases: Building a specific domain Ontology derived from database conceptual model. *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, pp. 1–8. URL: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5461774 (page 143).

Khentout, C., M. Djoudi, and L. Douidi (2007). Roundup of Graphical Navigation Helpers on the Web. *Journal of Computer Science* 3.3, pp. 154–161. ISSN: 1549-3636 (pages 10, 32, 90).

Kim, Harksoo, Choong-Nyoung Seon, and Jungyun Seo (2005). A Dialogue-Based Information Retrieval Assistant Using Shallow NLP Techniques in Online Sales Domains. *IEICE - Trans. Inf. Syst.* E88-D.5, pp. 801–808. ISSN: 0916-8532. DOI: 10.1093/ietisy/e88-d.5.801 (pages 9, 32, 89).

Kimura, Mikako and Yasuhiko Kitamura (2006). Embodied Conversational Agent Based on Semantic Web. *PRIMA*, pp. 734–741. DOI: 10.1007/11802372_87 (pages 9, 32, 90).

Klein, Dan and Christopher D. Manning (2002). Fast Exact Inference with a Factored Model for Natural Language Parsing. *In Advances in Neural Information Processing Systems 15 (NIPS)*. MIT Press, pp. 3–10 (pages 140, 143, 144).

Kuchmann-Beauger, Nicolas and Marie-Aude Aufaure (2011). A Natural Language Interface for Data Warehouse Question Answering. *Natural Language Processing and Information Systems*. Vol. 6716. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 201–208. ISBN: 978-3-642-22326-6. DOI: 10.1007/978-3-642-22327-3_21 (page 143).

Küçüktunç, Onur, Ugur Güdükbay, and Özgür Ulusoy (2007). A Natural Language-Based Interface for Querying a Video Database. *IEEE MultiMedia* 14.1, pp. 83–89. ISSN: 1070-986X. DOI: 10.1109/MMUL.2007.1 (page 141).

Ley, Michael (2002). The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. *String Processing and Information Retrieval*. Vol. 2476. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–10. ISBN: 978-3-540-44158-8. DOI: 10.1007/3-540-45735-6_1 (page 144).

Ley, Michael (2009). DBLP: Some Lessons Learned. *Proc. VLDB Endow.* 2.2, pp. 1493–1500. ISSN: 2150-8097. DOI: 10.14778/1687553.1687577 (page 144).

Li, Fei and H. V. Jagadish (2014a). Constructing an Interactive Natural Language Interface for Relational Databases. *Proceedings of the VLDB Endowment* 8.1. DOI: 10.14778/2735461.2735468 (pages 144, 189).

Li, Fei and H. V. Jagadish (2014b). NaLIR: An Interactive Natural Language Interface for Querying Relational Databases. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. SIGMOD '14. Snowbird, Utah, USA: ACM, pp. 709–712. ISBN: 978-1-4503-2376-5. DOI: 10.1145/2588555.2594519 (pages 13, 36, 144, 147, 189).

Li, Yunyao, Huahai Yang, and H. V. Jagadish (2005). NaLIX: an Interactive Natural Language Interface for Querying XML. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. SIGMOD '05. Baltimore, Maryland: ACM, pp. 900–902. ISBN: 1-59593-060-4. DOI: 10.1145/1066157.1066281 (pages 138, 141, 193, 199).

Li, Yunyao, Ishan Chaudhuri, Huahai Yang, Satinder Singh, and H. V. Jagadish (2007). DaNaLIX: a Domain-adaptive Natural Language Interface for Querying XML. *Proceedings of the 2007 ACM SIGMOD international conference*

on Management of data. SIGMOD '07. Beijing, China: ACM, pp. 1165–1168. ISBN: 978-1-59593-686-8. DOI: 10.1145/1247480.1247643 (page 141).

Lieberman, Henry (1995). Letizia: An Agent That Assists Web Browsing. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*. Ed. by Chris S. Mellish. Montreal, Quebec, Canada: Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, pp. 924–929. URL: http://www.aaai.org/Papers/Symposia/Fall/1995/FS-95-03/FS95-03-016.pdf (pages 8, 31, 86).

Lim, JongHyun and Kyong-Ho Lee (2010). Constructing composite web services from natural language requests. *Web Semantics: Science, Services and Agents on the World Wide Web* 8.1, pp. 1 –13. ISSN: 1570-8268. DOI: 10.1016/j.websem.2009.09.007 (page 143).

López, V., E. M. Eisman, and J. L. Castro (2008). A Tool for Training Primary Health Care Medical Students: The Virtual Simulated Patient. *ICTAI '08: Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence*. Vol. 2. Washington, DC, USA: IEEE Computer Society, pp. 194–201. ISBN: 978-0-7695-3440-4. DOI: 10.1109/ICTAI.2008.50 (pages 214, 230, 248).

López, Víctor, Eduardo M. Eisman, María Navarro, Jose Manuel Zurita, and Juan Luis Castro (2013). Un Paciente Simulado Virtual Multilingüe para la formación en medicina. *IE Comunicaciones: Revista Iberoamericana de Informática Educativa* 18, pp. 29–40. ISSN: 1699-4574. URL: http://dialnet.unirioja.es/servlet/articulo?codigo=4468676 (pages 214, 230, 250).

López Salazar, Víctor, Eduardo M. Eisman Cabeza, Juan Luis Castro Peña, and Jose Manuel Zurita López (2012). A case based reasoning model for multilingual language generation in dialogues. *Expert Systems with Applications* 39.8, pp. 7330 –7337. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.01.085 (pages 46, 249).

McDonald, Ryan, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee

(2013). Universal Dependency Annotation for Multilingual Parsing. *51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*. Vol. 2. Sofia, Bulgaria, pp. 92–97. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.386.8061&rep=rep1&type=pdf#page=140 (page 181).

Medina, Javier, Eduardo M. Eisman, and Juan Luis Castro (2013). Asistentes virtuales en plataformas 3.0. *IE Comunicaciones: Revista Iberoamericana de Informática Educativa* 18, pp. 41–49. ISSN: 1699-4574. URL: http://dialnet.unirioja.es/servlet/articulo?codigo=4468692 (page 250).

Micarelli, Alessandro and Filippo Sciarrone (1996). A case-based system for adaptive hypermedia navigation. *Advances in Case-Based Reasoning*. Lecture Notes in Computer Science. Springer, pp. 266–279. DOI: 10.1007/BFb0020616 (pages 9, 31, 87).

Miller, George A. (1995). WordNet: A Lexical Database for English. *Commun. ACM* 38.11, pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748 (pages 139, 141, 143, 195).

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller (1990). Introduction to WordNet: An On-line Lexical Database. *International journal of lexicography* 3.4, pp. 235–244. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.1244&rep=rep1&type=pdf (pages 139, 141, 143, 195).

Minock, Michael (2005). A Phrasal Approach to Natural Language Interfaces over Databases. *Natural Language Processing and Information Systems*. Vol. 3513. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 333–336. ISBN: 978-3-540-26031-8. DOI: 10.1007/11428817_30 (page 139).

Minock, Michael (2010). C-Phrase: A system for building robust natural language interfaces to databases. *Data & Knowledge Engineering* 69.3. Special Issue: 13th International Conference on Natural Language and Information Systems (NLDB 2008) - Five selected and extended papers, pp. 290 –302. ISSN: 0169-023X. DOI: 10.1016/j.datak.2009.10.007 (pages 142, 149).

Minock, Michael, Peter Olofsson, and Alexander Näslund (2008). Towards Building Robust Natural Language Interfaces to Databases. *Natural Language and*

Information Systems. Vol. 5039. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 187–198. ISBN: 978-3-540-69857-9. DOI: 10.1007/978-3-540-69858-6_19 (pages 142, 149).

Moreo, A., E.M. Eisman, J.L. Castro, and J.M. Zurita (2013). Learning regular expressions to template-based FAQ retrieval systems. *Knowledge-Based Systems* 53.0, pp. 108 –128. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2013.08.018 (pages 156, 251).

Mylopoulos, J., A. Borgida, P. Cohen, N. Roussopoulos, J. Tsotsos, and H. Wong (1976). TORUS: A step towards bridging the gap between data bases and the casual user. *Information Systems* 2.2, pp. 49 –64. ISSN: 0306-4379. DOI: 10.1016/0306-4379(76)90009-0 (page 132).

Nakano, Mikio, Yasuhiro Minami, Stephanie Seneff, Timothy J. Hazen, D. Scott Cyphers, James Glass, Joseph Polifroni, and Victor Zue (2001). Mokusei: a Telephone-based Japanese Conversational System in the Weather Domain. *INTERSPEECH*, pp. 1331–1334. URL: http://www.ai.mit.edu/projects/ntt/projects/MIT2001-05/documents/mokusei.pdf (pages 135, 203).

Nelken, Rani and Nissim Francez (2000). Querying Temporal Databases Using Controlled Natural Language. *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*. COLING '00. Saarbrücken, Germany: Association for Computational Linguistics, pp. 1076–1080. DOI: 10.3115/992730.992808 (page 136).

Noy, Natalya F., Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen (2001). Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems* 16, pp. 60–71. ISSN: 1541-1672. DOI: 10.1109/5254.920601 (pages 94, 153).

Ortony, A., G. L. Clore, and A. Collins (1988). The Cognitive Structure of Emotions. Cambridge University Press (pages 7, 29, 50).

Owda, M., Z. Bandar, and K. Crockett (2007). Conversation-Based Natural Language Interface to Relational Databases. *Web Intelligence and Intelligent Agent Technology Workshops, 2007 IEEE/WIC/ACM International Conferences on*, pp. 363–367. DOI: 10.1109/WI-IATW.2007.60 (page 141).

Pazos R., Rodolfo A., Juan J. González B., and Marco A. Aguirre L. (2011). Semantic Model for Improving the Performance of Natural Language Interfaces to Databases. *Advances in Artificial Intelligence*. Vol. 7094. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 277–290. ISBN: 978-3-642-25323-2. DOI: 10.1007/978-3-642-25324-9_24 (page 143).

Pazos R., Rodolfo A., Juan J. González B., Marco A. Aguirre L., José A. Martínez F., and Héctor J. Fraire H. (2013). Natural Language Interfaces to Databases: An Analysis of the State of the Art. *Recent Advances on Hybrid Intelligent Systems*. Vol. 451. Studies in Computational Intelligence. Springer Berlin / Heidelberg, pp. 463–480. ISBN: 978-3-642-33020-9. DOI: 10.1007/978-3-642-33021-6_36 (pages 11, 33, 145, 147–149).

Pazos Rangel, Rodolfo A., Joaquín Pérez O., Juan Javier González B., Alexander Gelbukh, Grigori Sidorov, and Myriam J. Rodríguez M. (2005). A Domain Independent Natural Language Interface to Databases Capable of Processing Complex Queries. *MICAI 2005: Advances in Artificial Intelligence*. Vol. 3789. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 833–842. ISBN: 978-3-540-29896-0. DOI: 10.1007/11579427_85 (page 139).

Pilato, Giovanni, Roberto Pirrone, and Riccardo Rizzo (2008). A KST-based system for student tutoring. *Applied Artificial Intelligence* 22.4, pp. 283–308. ISSN: 0883-9514. DOI: 10.1080/08839510801972785 (pages 10, 32, 91).

Plutchik, Robert (1960). The Multifactor-Analytic Theory of Emotion. *Journal of Psychology* 50, pp. 153–171. DOI: 10.1080/00223980.1960.9916432 (pages 48, 49, 53).

Popescu, Ana-Maria, Oren Etzioni, and Henry Kautz (2003). Towards a theory of natural language interfaces to databases. *Intelligent User Interfaces*, pp. 149–157. DOI: 10.1145/604045.604070 (pages 11, 34, 137, 138, 148, 189).

Prendinger, H. and M. Ishizuka (2001). Let's talk! Socially intelligent agents for language conversation training. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*. 31.5, pp. 465–471. DOI: 10.1109/3468.952722 (pages 7, 30, 52).

Ramachandran, Vivek Anandan and Ilango Krishnamurthi (2009). NLION: Natural Language Interface for querying ONtologies. *Proceedings of the 2Nd Bangalore Annual Compute Conference*. COMPUTE '09. Bangalore, India: ACM, 17:1–17:4. ISBN: 978-1-60558-476-8. DOI: 10.1145/1517303.1517322 (page 142).

Raux, Antoine, Dan Bohus, Brian Langner, Alan W. Black, and Maxine Eskenazi (2006). Doing Research on a Deployed Spoken Dialogue System: One Year of Let's Go! Experience. *in Proc. INTERSPEECH, 2006*, pp. 65–68. URL: http://www.cs.cmu.edu/~dbohus/docs/letsgo_interspeech2006.pdf (pages 140, 148).

Reis, Paulo, João Matias, and Nuno Mamede (1997). Edite - A Natural Language Interface to Databases A new dimension for an old approach. *Information and Communication Technologies in Tourism 1997*. Springer Vienna, pp. 317–326. ISBN: 978-3-211-82963-9. DOI: 10.1007/978-3-7091-6848-6_33 (pages 135, 149).

Rosis, Fiorella de, Catherine Pelachaud, Isabella Poggi, Valeria Carofiglio, and Berardina De Carolis (2003). From Greta's Mind to her Face: Modelling the Dynamics of Affective States in a Conversational Embodied Agent. *International Journal of Human Computer Studies* 59.1-2, pp. 81–118. DOI: 10.1016/S1071-5819(03)00020-X (pages 7, 30, 51, 52).

Ruiz-Martínez, Juana María, Dagoberto Castellanos-Nieves, Rafael Valencia-García, Jesualdo Tomás Fernández-Breis, Francisco García-Sánchez, Pedro José Vivancos-Vicente, Juan Salvador Castejón-Garrido, Juan Bosco Camón, and Rodrigo Martínez-Béjar (2009). Accessing Touristic Knowledge Bases through a Natural Language Interface. *Knowledge Acquisition: Approaches, Algorithms and Applications*. Vol. 5465. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 147–160. ISBN: 978-3-642-01714-8. DOI: 10.1007/978-3-642-01715-5_13 (page 142).

Russell, James A. and Lisa Feldman Barrett (1999). Core Affect, Prototypical Emotional Episodes, and Other Things Called Emotion: Dissecting the Elephant. *Journal of Personality and Social Psychology* 76.5, pp. 805–819. URL: http://emotiondevelopmentlab.weebly.com/uploads/2/5/2/0/25200250/russell_j.a.___barrett_l._f._1999.pdf (pages 7, 29, 48, 49).

Scha, Remko J. H. (1977). Philips question-answering system PHLIQA1. *Intelligence/sigart Bulletin* (61), pp. 26–27. DOI: 10.1145/1045283.1045291 (pages 133, 148).

Semeraro, Giovanni, Hans H. K. Andersen, Verner Andersen, Pasquale Lops, and Fabio Abbattista (2003). Evaluation and Validation of a Conversational Agent Embodied in a Bookstore. *Universal Access Theoretical Perspectives, Practice, and Experience*. Vol. 2615. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 360–371. ISBN: 978-3-540-00855-2. DOI: 10.1007/3-540-36572-9_28 (page 137).

Seneff, S. (2002). Response planning and generation in the MERCURY flight reservation system. *Computer Speech & Language* 16.3-4. Spoken Language Generation, pp. 283 –312. ISSN: 0885-2308. DOI: 10.1016/S0885-2308(02)00011-6 (pages 135, 189).

Seneff, Stephanie and Joseph Polifroni (2000). Dialogue Management in the Mercury Flight Reservation System. *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems - Volume 3*. ANLP/NAACL-ConvSyst '00. Seattle, Washington: Association for Computational Linguistics, pp. 11–16. DOI: 10.3115/1117562.1117565 (pages 12, 35, 135, 148, 189).

Seneff, Stephanie, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue (1998). GALAXY-II: a reference architecture for conversational system development. *ICSLP*. Vol. 98, pp. 931–934. URL: http://groups.csail.mit.edu/sls/publications/1998/icslp98-galaxy.pdf (page 135).

Seneff, Stephanie, Raymond Lau, James Glass, and Joseph Polifroni (1999). The MERCURY System for Flight Browsing and Pricing. *Laboratory for Computer Science - Spoken Language Systems. Massachusetts Institute of Technology*, p. 23. URL: http://groups.csail.mit.edu/sls/publications/1999/researchsummary99.pdf#page=39 (pages 135, 189).

Seneff, Stephanie, Chian Chuu, and D. Scott Cyphers (2000). ORION: from on-line interaction to off-line delegation. *INTERSPEECH*, pp. 142–145. URL: https://groups.csail.mit.edu/sls/publications/2000/03038.pdf (page 135).

Shankar, A and W Yung (2000). gNarLI: A practical Approach to Natural Language Interfaces to Databases. *Term Report, Harvard University* (pages 11, 34, 136, 146).

Sijtsma, Wietske and Olga Zweekhorst (1993). Comparison and review of commercial natural language interfaces. *Proceedings of the fifth Twente Workshop on Language Technology*, pp. 43–57. URL: http://eprints.eemcs.utwente.nl/9559/01/twlt5.pdf#page=44 (page 131).

Silverman, Barry G., Mintu Bachann, and Khaled Al-Akharas (2001). Do What I Mean: Online Shopping with a Natural Language Search Agent. *IEEE Intelligent Systems* 16.4, pp. 48–53. ISSN: 1541-1672. DOI: 10.1109/5254.941357 (page 136).

Sirin, Evren, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5.2. Software Engineering and the Semantic Web, pp. 51 –53. ISSN: 1570-8268. DOI: 10.1016/j.websem.2007.03.004 (page 141).

Sleator, Daniel D. K. and Davy Temperley (1991). Parsing English with a Link Grammar. Tech. rep. URL: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/sleator/public/link-grammar/LG-tech-report.pdf (page 139).

Staab, Steffen and Rudi Studer (2009). Handbook on Ontologies. Berlin: Springer. ISBN: 9783540926733 3540926739. DOI: 10.1007/978-3-540-92673-3 (page 154).

Stratica, Niculae, Leila Kosseim, and Bipin C. Desai (2005). Using semantic templates for a natural language interface to the CINDI virtual library. *Data & Knowledge Engineering* 55.1. Natural Language and Database and Information Systems NLDB 03, pp. 4 –19. ISSN: 0169-023X. DOI: 10.1016/j.datak.2004.12.002 (page 139).

Templeton, Marjorie (1979). EUFID: A Friendly and Flexible Front-End For Data Management Systems. *Meeting of the Association for Computational Linguistics*. DOI: 10.3115/982163.982190 (pages 133, 148).

Templeton, Marjorie (1983). Problems in natural-language interface to DBMS with examples from EUFID. *Applied Natural Language Processing Conference*, pp. 3–16. DOI: 10.3115/974194.974197 (pages 133, 148).

Thiel, U., M. L'Abbate, A. Paradiso, A. Stein, G. Semeraro, F. Abbattista, and P. Lops (2003). Intelligent E-Commerce with Guiding Agents Based on Personalized Interaction Tools. *E-Business Applications*. Advanced Information Processing. Springer Berlin Heidelberg, pp. 61–76. ISBN: 978-3-642-62846-7. DOI: 10.1007/978-3-642-55792-7_5 (page 138).

Thompson, Bozena H. and Frederick B. Thompson (1983). Introducing ASK, A Simple Knowledgeable System. *Proceedings of the first conference on Applied natural language processing*. ANLC '83. Santa Monica, California: Association for Computational Linguistics, pp. 17–24. DOI: 10.3115/974194.974198 (pages 134, 148).

Thompson, Bozena H. and Frederick B. Thompson (1985). ASK is Transportable in Half a Dozen Ways. *ACM Trans. Inf. Syst.* 3.2, pp. 185–203. ISSN: 1046-8188. DOI: 10.1145/3914.3983 (page 134).

Thompson, F. B., P. C. Lockemann, B. Dostert, and R. S. Deverill (1969). REL: A Rapidly Extensible Language System. *Proceedings of the 1969 24th National Conference*. ACM '69. New York, NY, USA: ACM, pp. 399–417. DOI: 10.1145/800195.805947 (pages 132, 148).

Váquez Granado, Javier, Rocío Ruiz, Ana Gonzáles, Carlos González, Javier Guerrero, Juan Luis Castro, Jose Manuel Zurita, Víctor López, Eduardo Eisman, and María Isabel Navarro (2008). Virtual Simulated Patient: A Novel Health Professionals' Training. *Proceedings of the 14th Annual Meeting of the Society in Europe for Simulation Applied to Medicine (SESAM 2008)*. Hatfield, The United Kingdom (pages 214, 230, 247).

Velásquez, Juan D. (1997). Modeling Emotions and Other Motivations in Synthetic Agents. *Proceedings of the 1997 14th National Conference on Artificial Intelligence, AAAI 97*. Ed. by Anon. Conference Code: 48599. Providence, RI, USA: AAAI, pp. 10–15. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.4894&rep=rep1&type=pdf (pages 7, 30, 52).

Wagner, Ferdinand, Ruedi Schmuki, Thomas Wagner, and Peter Wolstenholme (2006). Modeling Software with Finite State Machines: a practical approach. Boca Raton, FL. ISBN: 0849380863 (page 178).

Wallace, Richard S. (2003). The Elements of AIML Style. ALICE A. I. Foundation, Inc. URL: http://www.alicebot.org/style.pdf (pages 9, 32, 90).

Waltz, David L. (1978). An English Language Question Answering System for a Large Relational Database. *Commun. ACM* 21.7, pp. 526–539. ISSN: 0001-0782. DOI: 10.1145/359545.359550 (pages 11, 34, 133, 148).

Wang, Chong, Miao Xiong, Qi Zhou, and Yong Yu (2007). PANTO: A Portable Natural Language Interface to Ontologies. *The Semantic Web: Research and Applications*. Vol. 4519. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 473–487. ISBN: 978-3-540-72666-1. DOI: 10.1007/978-3-540-72667-8_34 (page 141).

Warren, David H. D. and Fernando C. N. Pereira (1982). An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *Computational Linguistics* 8.3-4, pp. 110–122. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972944 (pages 12, 35, 134, 149).

Wexelblat, Alan and Pattie Maes (1999). Footprints: History-Rich Tools for Information Foraging. *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*. Pittsburgh, Pennsylvania, United States: ACM, pp. 270–277. ISBN: 0-201-48559-1. DOI: 10.1145/302979.303060 (pages 9, 31, 88).

Wong, Yuk Wah and Raymond J. Mooney (2006). Learning for Semantic Parsing with Statistical Machine Translation. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. HLT-NAACL '06. New York: Association for Computational Linguistics, pp. 439–446. DOI: 10.3115/1220835.1220891. URL: http://dx.doi.org/10.3115/1220835.1220891 (pages 12, 35, 139, 148).

Woods, W. A., R. Kaplan, and B. Nash-Webber (1972). The LUNAR Sciences Natural Language Information System: Final Report. Final Report BBN Re-

port No. 2378. Cambridge, Massachusetts: Bolt, Beranek and Newman, Inc. (pages 11, 34, 132, 147).

World Wide Web Consortium (2004). OWL-S: Semantic Markup for Web Services. URL: http://www.w3.org/Submission/OWL-S/ (page 143).

Yan, Tak Woon, Matthew Jacobsen, Hector Garcia-Molina, and Umeshwar Dayal (1996). From user access patterns to dynamic hypertext linking. *5th International World Wide Web Conference*. Vol. 28. Paris, France, pp. 1007–1014. DOI: 10.1016/0169-7552(96)00051-7 (pages 8, 31, 87).

Yanaru, Tarao, Toyohiko Hirotja, and Naoki Kimura (1994). An emotion-processing system based on fuzzy inference and its subjective observations. *International Journal of Approximate Reasoning* 10.1, pp. 99 –122. ISSN: 0888-613X. DOI: 10.1016/0888-613X(94)90011-6 (pages 7, 29, 30, 48, 49, 53, 56).

Yates, Alexander, Oren Etzioni, and Daniel Weld (2003). A Reliable Natural Language Interface to Household Appliances. *Proceedings of the 8th International Conference on Intelligent User Interfaces*. IUI '03. Miami, Florida, USA: ACM, pp. 189–196. ISBN: 1-58113-586-6. DOI: 10.1145/604045.604075 (page 138).

Zambetta, F., G. Catucci, F. Abbattista, and G. Semeraro (2003). SAMIR: Your 3D Virtual Bookseller. *Knowledge-Based Intelligent Information and Engineering Systems*. Vol. 2773. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1249–1257. ISBN: 978-3-540-40803-1. DOI: 10.1007/978-3-540-45224-9_169 (page 138).

Zambetta, Fabio and Fabio Abbattista (2005). The Design and Implementation of SAMIR. *Knowledge-Based Intelligent Information and Engineering Systems*. Vol. 3682. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 768–774. ISBN: 978-3-540-28895-4. DOI: 10.1007/11552451_105 (page 138).

Zhang, Dong-Mo, Huan-Ye Sheng, Fang Li, and Tun-Fang Yao (2002). The model design of a case-based reasoning multilingual natural language interface for database. *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*. Vol. 3, 1474–1478 vol.3. DOI: 10.1109/ICMLC.2002.1167452 (page 137).

Zhang, G., W. W. Chu, F. Meng, and G. Kong (1999). Query formulation from high-level concepts for relational databases. *User Interfaces to Data Intensive Systems, 1999. Proceedings*, pp. 64–74. DOI: 10.1109/UIDIS.1999.791463 (pages 136, 148).

Zhou, Lina, Ammar S. Mohammed, and Dongsong Zhang (2012). Mobile personal information management agent: Supporting natural language interface and application integration. *Information Processing & Management* 48.1, pp. 23–31. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2011.08.008 (pages 2, 24, 128).

Zue, V., S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington (2000). JUPITER: a telephone-based conversational interface for weather information. *Speech and Audio Processing, IEEE Transactions on* 8.1, pp. 85–96. ISSN: 1063-6676. DOI: 10.1109/89.817460 (pages 12, 35, 135, 148, 189).

Zue, Victor, James Glass, David Goodine, Hong Leung, Michael Phillips, Joseph Polifroni, and Stephanie Seneff (1992). The Voyager Speech Understanding System: A Progress Report. *Speech Recognition and Understanding*. Vol. 75. NATO ASI Series. Springer Berlin Heidelberg, pp. 415–424. ISBN: 978-3-642-76628-2. DOI: 10.1007/978-3-642-76626-8_41 (page 135).

Zue, Victor, Stephanie Seneff, Joseph Polifroni, Michael Phillips, Christine Pao, David Goddeau, James Glass, and Eric Brill (1994). PEGASUS: A Spoken Language Interface for On-line Air Travel Planning. *Proceedings of the Workshop on Human Language Technology*. HLT '94. Plainsboro, NJ: Association for Computational Linguistics, pp. 201–206. ISBN: 1-55860-357-3. DOI: 10.3115/1075812.1075855 (page 135).

Zue, Victor, Stephanie Seneff, James Glass, Lee Hetherington, Edward Hurley, Helen Meng, Christine Pao, Joseph Polifroni, Rafael Schloming, and Philipp Schmid (1997). From interface to content: translingual access and delivery of on-line information. *EuroSpeech*. URL: http://groups.csail.mit.edu/sls/publications/1996/eurospeech97-jupiter.pdf (pages 135, 189).

Universidad de Granada