Natural Language Techniques for Model-Driven Semantic Constraint Extraction

Marita Ailomaa, Martin Rajman
Artificial Intelligence Laboratory (LIA)
School of Computer and Communication Sciences (I&C)
École Polytechnique Fédérale de Lausanne (EPFL)
CH-1015, Lausanne, Switzerland
{marita.ailomaa, martin.rajman} @epfl.ch

July 7, 2005

EPFL Technical Report

Abstract

This report describes a model-driven approach to natural language understanding (NLU) in which the "meaning" of natural language queries is extracted based on a domain model composed of a set of concepts and relations specified in the system's domain. The extracted meaning is represented as a set of semantic constraints that describe concept instances and their relations. This representation can easily be translated into a form usable for the system, e.g. an SQL-query or a dialogue move. The method has been tested on a dialogue system for browsing and searching in a database of recorded multimodal meetings, and experimental results as well as a discussion on the robustification of the approach are provided.

Keywords: Natural language understanding, syntactic analysis, logical forms

Contents

Chapter 1
Introduction

1.1. Natural language understanding

Natural language understanding (NLU) is the field within computational linguistics and computer that deals with machine understanding of human language. A large number of approaches have been explored to extract and represent the "meaning" of natural language sentences, with the goal to enable computers to interact with humans in their own language (Allen, 1995). The motivation for this type of intelligence in computers is that, up to now, the most common computer applications are programs, where the user interacts with the system by clicking on buttons, selecting elements in menus or typing commands. It has been argued that this is not the most natural way for humans to interact, especially if the user is a *non-expert* user of the system (Allen et al. (2001)). Such systems are often designed to provide services that are otherwise handled by human-experts, for instance travel agents, receptionists and information staff. When a system takes over the task, either the user has to learn to use the functionalities of the system (which a human-expert is skilled to do), or the system can try to *simulate* the human-expert, so that users can continue interacting in the way that is most intuitive to them – in natural language.

Some of the most typical applications of NLU are dialogue systems, for instance travel agency services, train and bus schedule services and systems for commanding home devices (Rudnucky et al. (1999), Larsson et al. (2000)). Another widely studied area of NLU is Natural Language Interfaces to Databases (NLIDB) (Androutsopoulos et al. (1996)). Such interfaces relieve the users of the need to know the structure of the database and offer a much more convenient and flexible interaction, allowing queries to be expressed in natural language, e.g. *"Show the domestic suppliers"* or *"What are the names of our Canadian customers?"*. A more recent field in NLU is Question Answering (QA) which differs from NLIDB in one important aspect. QA systems try to answer fact-based questions about any topic (e.g. *"Who was the prime minister of Canada in 1873?"*) by analyzing a collection of unrestricted and unstructured texts (Harabagiu et. Al (2000), Zheng et al. (2002)). The approaches used in QA can however be very useful for NLIDB.

In all application fields, NLU is a difficult problem that requires processing of the natural language input in several steps. If a system accepts vocal input, the utterance first has to be recognized by an automatic speech recognizer (ASR) and transcribed into text. The main problems in speech recognition are that users have individual voices and

pronunciations and that spoken language is often disfluent, which might lead to many recognition errors.

Once the natural language input has been transcribed into text, a syntactic analysis is often important to verify that the sentence is well-formed and to add structural information to the sequence of words. Lexicons and grammars are resources used at this level of analysis, and some of the main problems that have been addressed in the area are ambiguity, out-of-vocabulary words and undergeneration (sentences that are beyond the scope of the grammar).

Both speech recognition and syntactic analysis can be performed with application-independent algorithms using large, wide-coverage linguistic resources. Many systems are capable of processing language at these two levels with very good performance (Cole et al. (1997)). Semantic analysis, on the other hand, is strongly dependent on the application, and there seems to exist far less agreement about the methods for extracting and representing meaning (Wahlster (2000)). The techniques for "understanding" can range from simple keyword spotting to formalizations in first-order logic and complex logical reasoning. In the next section, we give a definition of natural language understanding that is applicable in real situations, not attempting to formally represent the linguistic meaning of utterances, but instead focusing on the system and the task that it is designed for and narrowing down the notion of "meaning" to what is meaningful for the application.

1.2 Natural language understanding from a model-driven perspective

The approach to natural language understanding, presented in this report, is based on a fact that applies to a wide range of natural language application found today: the domain is well-specified, which means that the knowledge that the system has about the world is of a fixed size (e.g. travel agency, train schedule etc.). This domain knowledge can be represented as a model of concepts and relations, which the user makes reference to in natural language. From the system perspective, it is only worthwhile to "understand" queries that match something in the domain model, hence a model-driven approach to natural language understanding. But one can also expect from the user to cooperate with the system, i.e. to ask questions that the system is able to answer. Since applications are created for specific tasks, the user should be aware, at least globally, of what the system is able to understand and what it is not.

The "understanding" of natural language queries is, in this sense, nothing more than a mapping between the linguistic form of the query and a domain-specific representation that the system is able to use for performing computational tasks. For instance, a natural language query can be transformed into an SQL-like form, from which a precise SQL query can be formulated and then executed on a database. In a dialogue system, moreover, the utterances are strongly contextualized during the dialogue, which should be an additional help for the NLU. In this report, we will describe a NLU component that maps natural language queries to a domain-specific form made of semantic constraints, an intermediate form easily transformable to an SQL-query or a dialogue move, depending on the target application. The NLU is based on two types of mapping rules

that use lexical and syntactic information (provided in the earlier processing phase), to extract a meaning representation that describes concepts and relations in the application domain.

The rest of this report is divided into six parts. Chapter 2 briefly describes an application where model-driven NLU is implemented. The goal is to give an overview of the main components of the system, to help the reader to understand the examples in the subsequent chapters. Chapter 3 introduces semantic constraints and conceptually describes how these map to the different elements in the natural language query. In chapter 4 we propose two architectures for processing queries through the different natural language processing modules (ASR, syntactic parser and NLU). The architectures are based on different speech recognition-paradigms and have impact on the design and implementation of the NLU module. Details about the linguistic resources and the syntactic representation (logical form) are provided. In chapter 5 we give the concrete implementation of the NLU module, following the first of the two paradigms. Experimental results are discussed in chapter 6, and issues related to the integration of the NLU module in the second architecture are brought up in chapter 7. Finally, chapter 8 concludes and gives directions for future work.

Chapter 2
Archivus: A multimodal meeting retrieval and browsing system

To give some background to how a model-driven approach to natural language understanding can be applied within a system, a starting point is to describe such a system. This chapter briefly presents Archivus, a dialogue system for browsing and searching through a database of recorded multimodal meetings. The focus of the chapter is on the parts of the system components that are relevant for the NLU. For more information about Archivus, the interested reader is refereed to Lisowska et al (2004).

Figure 1: The graphical user interface in Archivus

2.1 Interaction modalities

One of the key aims of Archivus is to provide a flexible and consistent access to different interaction modalities so that the users can choose to interact with the system in the way that they find most comfortable. This means that the user should be able to change between one modality and another at any time, if the one being used is not yielding the required results. The modalities considered in Archivus are voice, pointing (either through a mouse or a touch screen) and text.

There are in fact two distinct cases of natural language interaction; system commands and system queries. Vocal commands to the interface involve the control and manipulation of the visual elements. The more complex case occurs when the user provides natural language queries to the system, such as "*What decisions were made during the May 10th meeting?*", expecting to be given back a segment of a transcribed meeting that gives the answer to the question. This report focuses on this second case.

2.2 Data annotation

The data that the users access when searching for the contents of meetings is primarly the transcriptions of the recorded meetings. To enable a content-based search within the transcriptions (for instance a search for segments where decisions are made), the data is annotated at several levels. On utterance level the different types of annotations are *speaker identification*, *dialogue acts* and *utterance keywords*.

The second level of annotation is called *argumentative annotation*. It groups together a sequence of utterances into an argumentative *segment,* which represents an argumentative *class*, for instance *discuss(topic), discuss(issue), propose, reject, accept and decide*. This level of annotation is based on a structured description that defines the hierarchical structure of argumentation in meetings. One argumentative segment can embed other argumentative segments, which in turn can embed even smaller segments. An example of argumentative annotation is given in figure 2. (The horizontal lines represent the time dimension.) More details about the argumentative annotation can be found in Pallotta et al. (2003).

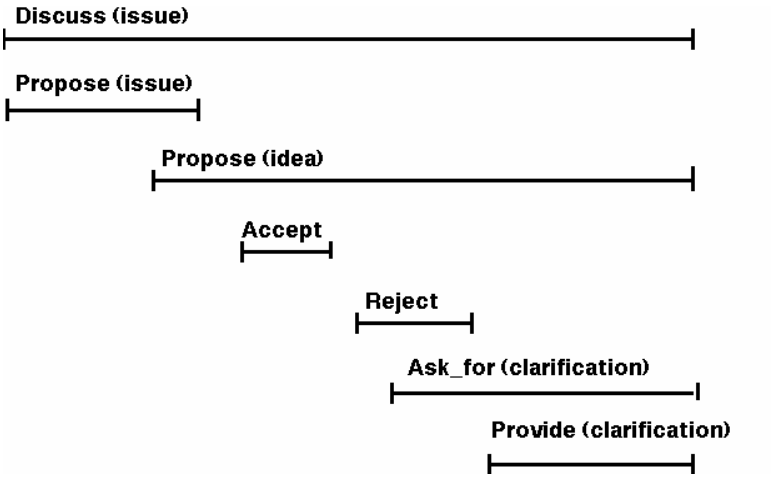

Figure 2: Argumentative annotation

The largest segments in the meeting transcriptions are *thematic episodes*, which are annotated with topics and keywords that summarize the content of large fractions of the meeting. Furthermore, documents presented or referred to during the meeting, and audio/video recordings, are linked to the parts of the transcriptions where they occur in time.

2.3 Data model

The relational dataschema of the database, that stores the recorded meetings, is specified through a data model. This model consists of concepts, attributes and relations that represent the annotations previously described (see Figure 3). The main concepts are *utterance*, *meeting*, *location, person, document, argumentative segment* and *thematic episode*. Each concept is characterized by a set of attributes. For instance a person has a *first name*, *family name* and *function*. Among the different relations between concepts, we find that a location can be the *address* of a person or the *place* of a meeting. Similarly, a person can be a *participant* of a meeting, a *speaker* of an utterance, or an *author* of a document.

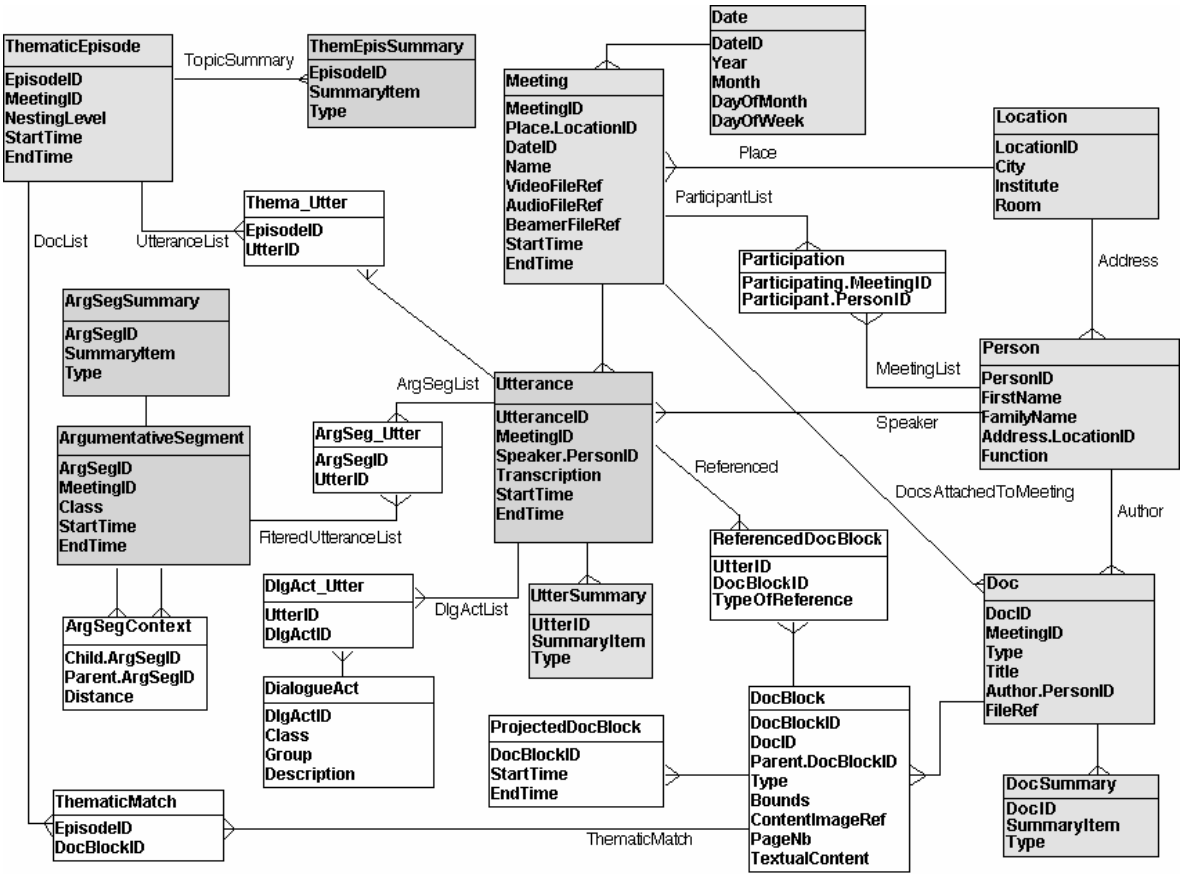

Figure 3: Data model

The four concepts *utterance*, *argumentative segment, thematic episode* and *document* all have a relation to a *summary* concept called *utterance summary*, *argumentative segment summary* and so on. These summary concepts represent the content of the different episodes and smaller segments of the meeting (or a document) through a set of *keywords* and *topics*, which are two types of *summary items*. The main difference between keywords and topics is that the keywords are actual words occurring in the dialogue of the meeting whereas the topics describe the content on a conceptual level and do not necessarily occur explicitly in the dialogue. *'Budget'* is an example of a topic that can be discussed without using the word *"budget"* (but instead using words like *"money"*, *"francs"*, *"finance"* etc.).

2.4 Generic dialogue nodes (GDNs)

When the user is making a search in Archivus, their interaction with the system is a dialogue, during which the goal is to find a meeting or a specific part of a meeting that contains the information that the user is interested in. The system asks the user to give values for different attributes such as the place of the meeting and the names of the participants. The simple interaction to obtain one value for one attribute is handled by a generic dialogue node (GDN) (Bui et al. (2004)). Figure 4 shows the different GDNs managed by Archivus. The more values the user provides, the more precise are the results that the system is able to return. Furthermore, the dialogue is mixed-initiative, so the user can at any time provide more (or other) values than the one asked for by the currently active GDN.

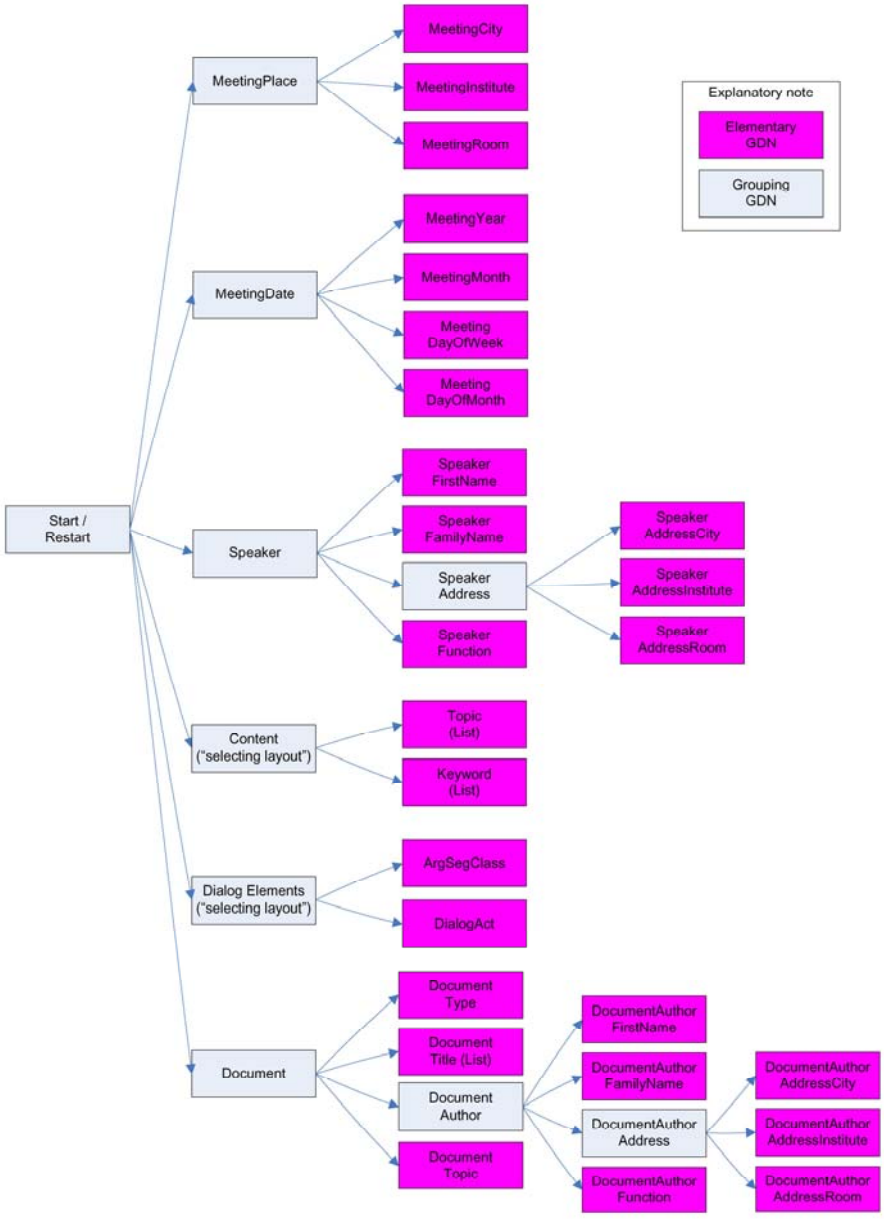

Figure 4: Generic dialogue nodes

Chapter 3
Extracting semantic constraints from user queries

When a natural language query is analyzed by the system, the goal is to make the mapping between the linguistic terms in the query and the concepts and relations existing in the system's domain. For instance, the query "*What decisions were made during the May 10th meeting?"* can be mapped to three concepts in the Archivus domain. "*Decisions"* are a class of argumentative segments in the annotation of a meeting. "*May 10th"* is a date. More precisely it is the *month* and the *day of month*, which are two attributes characterizing the concept *date.* Finally, the linguistic term "*meeting"* simply maps to the domain-specific concept *meeting*. The conceptual relations existing between these three concepts are that *"May 10th"* is the *date* of a *meeting* and that the *"decision"* was made at that same *meeting*.

3.1 Semantic constraints

How the domain-specific form of the query is represented, varies from application to application. A general approach is to first produce an intermediate form and then to derive the target representation from that, for instance an SQL-query (Rayner (1993), Androutsopoulos et al. (1996)). In Archivus, instead of trying to translate the intermediate form to SQL, the ultimate goal is to produce *attribute-value pairs* that feed the dialogue manager. We call the set of attribute-value pairs produced for a given query the *semantic constraints* set.

A semantic constraint is an element that describes a concept or a relation between two concepts. In our case, the constraint aims at identifying possible instances of a concept by eliminating the ones that violate the constraint.

3.2 Instances of concepts

A concept is described through its attributes or though its relation to other concepts. A semantic constraint therefore consists of an *instance identifier*, an *attribute* and a *value*, or an *instance identifier*, *relation* and another *instance identifier*. For instance, *"the May 10th meeting"* corresponds to three semantic constraints:

(1) date1.month = 'may'
(2) date1.dayofmonth = '10'
(3) meeting1.date = date1

For complex queries the mapping to semantic constraints can produce several instances of the same concept. It is also possible that one instance of a concept has several relations with other concepts. For instance, in Archivus, a complex query is:

(4) *Who rejected Agnes proposal about a coffee machine?*

"Who" and "*Agnes"* are two instances of the concept *person*, "*rejected"* and "*proposal"* refer to the argumentative classes '*reject'* and *'propose'*, and *"coffee machine"* is a *summary item*. The semantic constraints describing these concepts are:

(5) who ↑ person1
(6) Agnes ↑ person2.firstname = 'Agnes'
(7) rejected ↑ argseg1.class = 'reject'
(8) proposal ↑ argseg2.class = 'propose'
(9) coffee machine ↑ summary1.item = 'coffee machine'

Two special cases, worth a mention, are (5) and (9). The word "*who"* refers to a person, and although it does not specify which person it is (it does not impose any constraint on the instance of a person), it provides information that can be relevant when extracting relations between concepts. Therefore, it does not trigger a semantic constraint but only a *concept instance*, which does not have an attribute.

The second special case concerns *summary items*. According to the data model, a *summary item* describes a particular type of *summary*: *utterance summary*, *document summary* etc. Since the summary item itself does not tell which type of summary it belongs to *("coffee machine"* can be the summary item of a document as well as an utterance, argumentative segment or thematic episode), we create a general class called *summary*, which represents all four of them. The context of the item in the query will then decide which precise type of summary the item belongs to.

3.3 Relations between concepts

How the relations between the concepts are extracted is more complicated. Returning to query (4), one can see that there is a relation between *persons* and *argumentative segments*, for instance between "*Agnes"* and "*proposal"*. But in the data model this relation is implicit. A person can be a *speaker* of an *utterance*, and the utterance in turn occurs within the *argumentative segment*. So the semantic constraints produced by this relation involve three concepts: *person*, *argumentative segment* and *utterance*. "*Who rejected"* and "*Agnes' proposal"* give the following relational constraints, in addition to (5)-(9):

(10) argseg1.utterance = utterance1
(11) utterance1.speaker = person1
(12) argseg2.utterance = utterance2
(13) utterance2.speaker = person2

The second type of relation that exists in query (4) is between the two argumentative classes '*propose'* and '*reject'*. According to the argumentative model, a proposal can be responded by rejections and acceptances. To make explicit that the rejection (7) responds to the proposal (8), and not to any arbitrary proposal, we add the semantic constraint:

(14) argseg1.responds_to = argseg2

Finally, there is a relation between the argumentative segment (8) and the summary item (9). This is expressed by the semantic constraint:

(15) argseg2.summary = summary1

This final constraint disambiguates the type of summary that '*coffee machine'* belongs to. An overview of the complete example is given in figure 5.

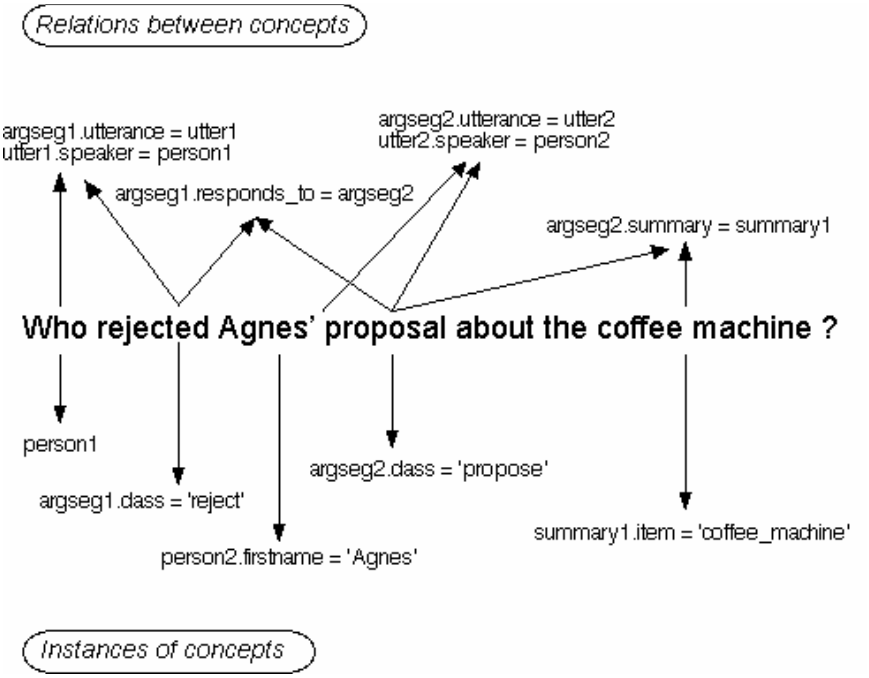

Figure 5: Semantic constraints

Chapter 4
An NLP architecture with grammar-based speech recognition

This chapter describes the NLP architecture, consisting of several components, in which semantic constraint extraction is integrated. First, two possible architectures are discussed. Then more background on the chosen architecture is given. The speech recognizer and the resources used for the speech recognition are very briefly introduced. Also the representation of the speech recognition output is explained, and how this representation (logical form) is post-processed into a flat quasi-logical form to facilitate the access to the parts that are relevant for the NLU module. We then describe how the semantic lexicon, used by the speech recognizer, is extended to provide information that aids the NLU module in identifying the words in the query that have a mapping to domain-specific concepts in the system.

4.1 Two architectures for a natural language understanding engine

In the previous chapter, we described how natural language queries are mapped to domain-specific representations called semantic constraint sets, a form easily translatable to SQL or to attribute-value pair sets used by the dialogue manager in Archivus. The goal of this report is to describe how the required semantic constraint extraction is performed automatically. For the semantic constraints to be extracted, the user queries need to undergo several levels of analysis: speech recognition, syntactic analysis and semantic analysis. The natural language processing component handling the different levels of analysis, is composed of three modules; the automatic speech recognizer, the syntactic analyzer and the NLU module. It relies on a sequential pipeline in which the results of one module are transmitted to the next. The precise nature of the results produced by each module, vary depending on the choice of the global architecture. This will also have an impact on the robustness of the system and on the correctness of the final output produced by the NLU module. We consider two possible architectures for the component, based on two ways of handling the speech recognition and its results. In the first architecture (see Figure 6) a grammar-based speech recognizer (Nuance) is used to produce 1 or *n*-best transcriptions (with *n* typically ranging from 5 to 10), represented as sequences of words and logical forms. (A detailed description of the logical form will be given in section 4.3.) The logical form(s) is (are) then passed on to the NLU module that performs semantic constraint extraction using the semantic and syntactic information, present in the logical form.

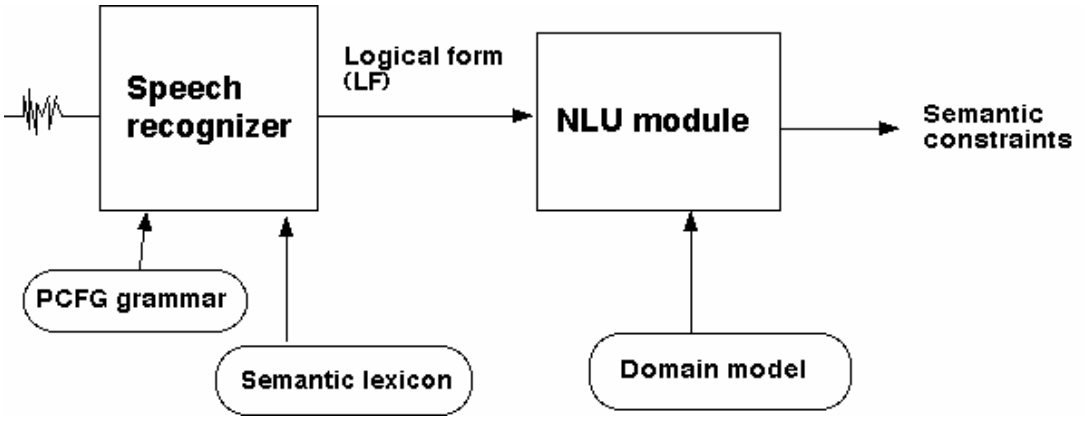

Figure 6: NLP architecture with grammar-based speech recognition (Architecture 1)

Grammar-based speech recognition has proved to be more reliable than robust speech recognition based on simpler language models such as n-grams, provided however that the user is familiar with the limited vocabulary understood by the system and with the type of utterances that the system is able to analyze with its grammatical resources (Knight et al. (2001)). The weakness of this approach is that the system can only recognize natural language queries that are fully described by the grammar, i.e. queries for which there is a derivation tree that covers the whole query The way humans use natural language is very complex, so it can be hard to write a grammar that covers all, or even most, of the input sentences in the application.

For the above reason, we also consider a robust approach to speech recognition. The second architecture is based on a speech recognizer that produces a word-lattice as output (see figure 7). A word-lattice is a compact representation of alternative transcriptions of the vocal input. It is a directed graph where every path though the graph is a possible word sequence.

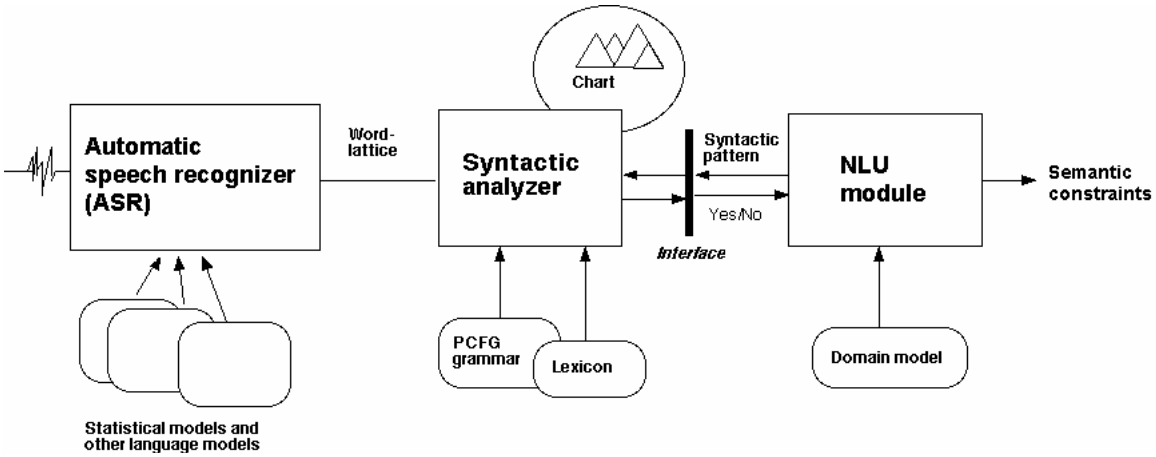

Figure 7: A robust NLP architecture (Architecture 2)

The syntactic analyzer parses the word-lattice with a context-free grammar and stores the resulting forest of derivation trees in a chart. Depending on the size of the word-lattice and the grammar, the forest can contain hundreds, thousands or even millions of trees, which makes it infeasible to unfold the chart into distinct trees. Instead the NLU module has to operate directly on the chart through a specific interface that provides the possibility to check whether derivation trees matching specific *syntactic patterns* are present in the chart. The goal is that, for a given user query, the NLU module should return the same set of semantic constraints, regardless of the chosen architecture.

This report mainly focuses on the first architecture and describes in some detail how user queries are processed through the different modules, concentrating on the mechanisms involved in the NLU.

4.2 Grammar-based speech recognition with Regulus and Nuance

The grammar-based speech recognizer, chosen for the current architecture of Archivus, is implemented with the Regulus and Nuance software. Regulus is a Prolog-based toolkit for developing unification-based grammars and semantic lexica. The syntactic analysis of a sentence, parsed with a Regulus-grammar and lexicon, can be represented both as a syntactic tree and as a logical form.

Regulus provides tools for compiling a unification-based grammar and lexicon into a grammar in the Nuance Grammar Specification Language (GSL), which is then used to build the corresponding Nuance speech-recognizer. To reduce the size of the GSL grammar, which can become extremely large if a very general Regulus-grammar, specialization is required. Regulus provides a tool for this, implementing an Explanation Based Learning method.

The Nuance speech-recognizer analyzes the vocal input using its grammar- and lexicon-resources and at the same time parses the produced transcription. The recognition result therefore consists of both a sequence of words and a logical form (the syntactic analysis of the query). With the current choice of parameters, the recognizer returns only the 1-best result, but it is also possible to get the *n*-best results, with values of *n* ranging from 5 to 10.

4.3 Logical forms

The logical form, provided as a result of the speech recognition, can be represented in several ways: linear forms, nested logical forms or RIACS logical forms. Without going into detail on the differences between these different representations, we choose the nested logical form, because it provides an adequate level of detail/abstraction of the available syntactic information. Figure 8 shows an example of this type of logical form. The nested logical form is a hierarchical syntactic analysis of the natural language query but not equivalent to the “standard” derivation trees that would be produced by traditional rule-based rewriting grammars. Logical forms are more abstract, focusing on the syntactic roles of the main elements in the sentence: the subject, main verb, object etc.

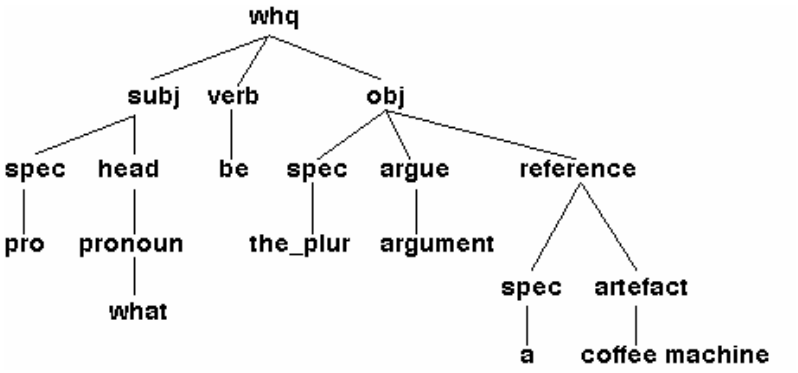

Figure 8: Nested logical form of "What were the arguments against a coffee machine?"

Some words in the sentence are not explicitly present in the logical form. Prepositions, for instance, are translated into syntactic roles. In the example in figure 8, the word *"against"* is represented as the syntactic role *reference*. Another difference is that the order of the elements in the logical form is not dependent on the order of the words in the query. Two sentences having different word orders can be represented by the same logical form. Typical examples are active and passive constructions, as illustrated in figure 9.

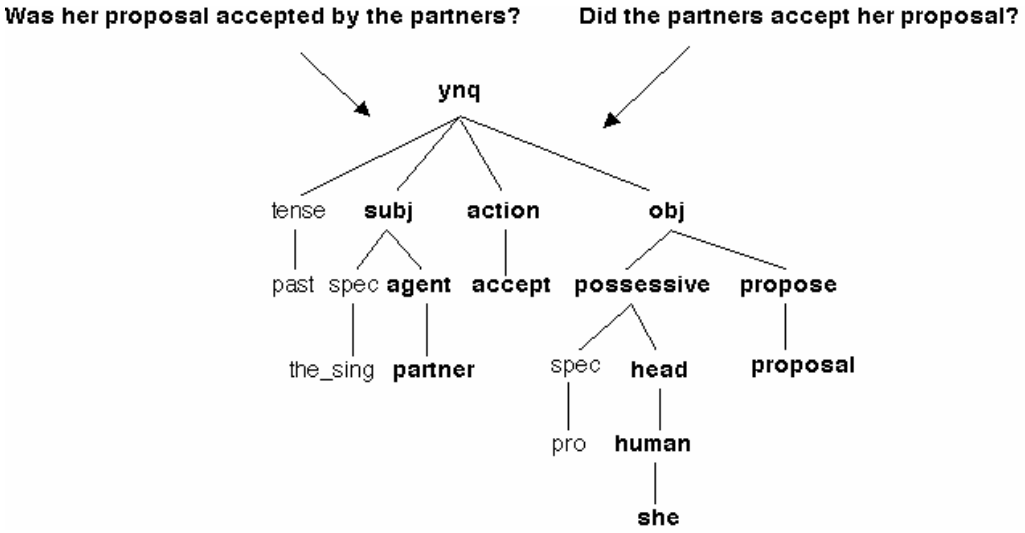

Figure 9: Logical form representing both active and passive construction

Besides the syntactic roles, the logical form also provides semantic information about the main words, for instance [*human*,she]. This information is expressed in terms of *semantic types*. The semantic types are specified in the lexicon, but they do not have any impact on the syntactic analysis of the query. They only provide a "semantic decoration" to the logical form. This "decoration" is however very useful for the NLU module. It helps to identify words in the query that are important for the application. The next section will describe how these semantic types are chosen for the lexicon in Archivus.

As the logical form is rather rich on syntactic and semantic information, the NLU module does not need to access the whole information to extract semantic constraints. To simplify the access to the relevant parts, the nested logical form is first transformed into a *flat quasi-logical form* (QLF) (see figure 10), and then processed by the NLU module.

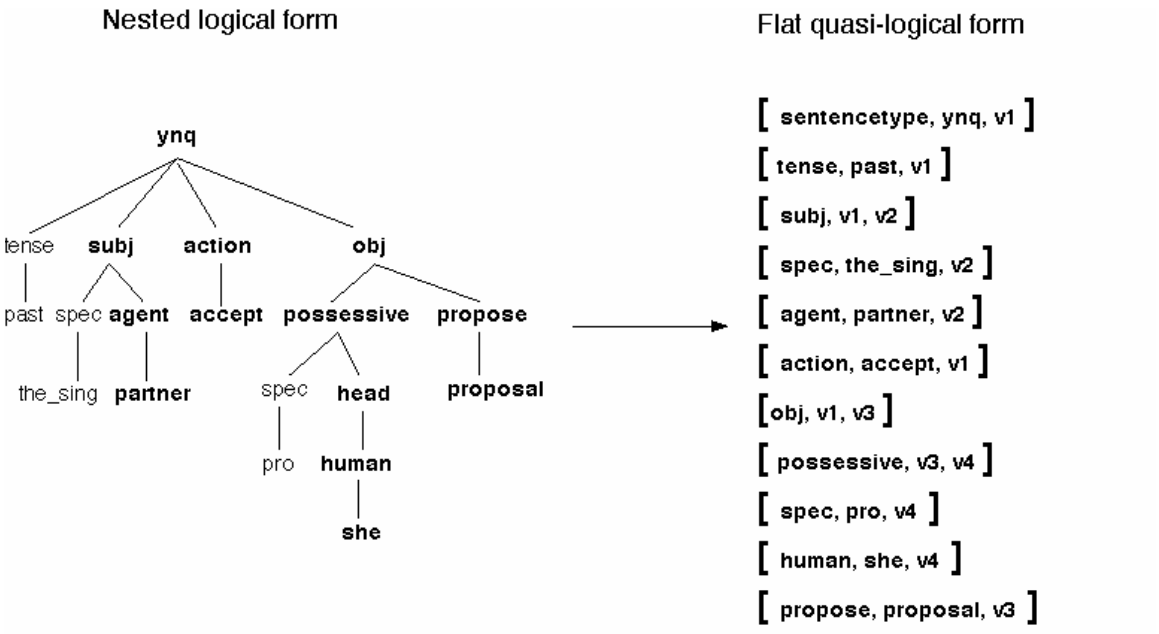

Figure 10: Tranformation from nested logical form to flat quasi-logical form (QLF)

In the flat QLF, each item corresponds to one node in the nested representation. In order to preserve the information about the dependencies between the nodes, a variable is added to each item, to represent its scope in the hierarchical structure. For instance, between the three items *sentencetype*, *subject* and lexicalized semantic item

[sentencetypee,ynq,v1]
[subj,v1,v2]
[human,who,v2]

the variables *v1* and *v2* inform that the subject is within the scope of the sentencetype and that the lexicalized semantic item *[human,who]* is the subject. In the rest of the paper, logical forms will be presented in the flat QLF notation.

4.4 The semantic lexicon

A semantic lexicon typically organizes words according to their meanings. There are two uses of a semantic lexicon in the natural language processing of queries in Archivus. The first use is within the syntactic analysis. The grammar uses lexical information to restrict the possible combinations of words in the syntactic structures. For instance, only a human entity can be the subject of certain actions such as reading, presenting, talking etc. With

such lexicalized semantic information the grammar can successfully analyze *"Which participants asked questions?"* while rejecting *"Which documents asked questions"*. This type of lexical information is called *sortal types* and is particularly useful during speech recognition, to prevent invalid transcriptions.

The second type of lexical information used in Archivus, are the semantic types appearing in the logical form. These types also classify words according to their meaning, but they have no impact on the syntactic analysis and can be chosen according to the needs of the application. Either the semantic types can describe the application-specific meaning of the words, e.g. *[argseg_class,disagree]* or they can represent the linguistic meaning, e.g. *[react,disagree]*. Since the logical form is an abstract *linguistic* representation of the query and the semantic lexicon is mainly used by the speech recognizer a modular approach would be to let the semantic types represent the linguistic meaning of the words rather than the domain-specific meaning. However, these semantic types have to provide the appropriate information to enable the NLU module to map words to domain-specific concepts, so the goal is to find a level of linguistic meaning that makes the mapping possible.

A large amount of work has already been done to build semantic dictionaries that organize words into conceptual hierarchies according to their meanings. WordNet (Fellbaum (1998)) is a lexical reference system that was designed as a network, to reflect how speakers organize their mental lexicons. Synonyms are grouped together into synsets representing concepts, and concepts are linked to one another through different relationships such as hyponymy (more specific meaning) and hypernymy (more general meaning).

WordNet is a well-known resource used in numerous projects and large enough to cover many domains through a large variety of synonyms for each word. We choose WordNet in an attempt to classify our list of words into general classes of concepts, though another larger semantic resource, EDR (Yokoi (1995)), might be used later.

The classification is done in the following way:
a) First the list of verbs and nouns are extracted from the available corpus of example user queries that are used as a basis for building the grammatical resources for the speech recognizer.
b) Second the list of verbs is nominalized in order to obtain the corresponding noun. These nouns are grouped with the verbs into the same conceptual class in the case where they can be considered as the result of the action expressed by the verbs, e.g. agree-agreement, approve-approval.
c) The next step is to identify the synsets in WordNet to which the verbs and nouns belong. This creates a small ontology, a sub set of the WordNet ontology that only contains words from the Archivus domain.
d) Finally, the ontology needs to be simplified by reducing the number of nodes. A word can belong to several synsets that represent different meanings of the word. Only the synsets relevant for the domain are kept. The reduction is done manually by peering into the classification of the WordNet concepts and retrieving the concepts that play a role in

the mapping to the Archivus domain. Choices are made about the specificity of the meaning of words. For instance, a path in the conceptual hierarchy is:

(16) accept < react,respond < act,move

Two nodes in the path represent concepts that are relevant in the domain: *accept* maps to the argumentative class "accept", *react/respond* maps to one of the argumentative classes "accept" or "reject" but doesn't specify which one.

Another example of relevant paths in the conceptual hierarchy are:

(17) elect < {choose, take, select} < {**decide**, make up one's mind, determine}
(18) **decide**<{determine,shape,mold,influence}<{cause,do,make}<{make,create}

Both meanings of the word *"decide"* are considered as relevant because they both can be mapped to the argumentative class *'decide'* in the domain model. But there cannot be two meanings for the word in the semantic lexicon, so we merge the two synsets into one concept *decide*. The nodes which are sub-concepts of *decide* in the first path (*choose*, *elect* etc.) are not kept because they are too specific to have a purpose in the Archivus domain. There is no annotation in the meeting records that allows the distinction between the two queries: *"Was there an election?"* and *"Was any decision made?"*. Both queries look for a fragment of a meeting that is annotated as an argumentative segment of class *'decide'*. In our semantic lexicon words that, according to WordNet, belong to synsets that are more specific than *decide* are all categorized within the same semantic type *decide*. Similarly, we merge the two synsets {*cause, do, make}* and *{create, make}* to one concept *create*. And the node *{determine, shape, mold, influence}* does not contain new words that are important in our domain, so this node is removed. In the end, we only have two nodes from the two initial paths representing two new concepts, *decide* and *create*:

(19) {decide, determine, choose, take, select make up one's mind} < {make, create, cause, do}

After having described how the grammar and lexical resources have been tuned to provide a syntactic analysis useful for the NLU module we move in the next chapter to the core of the work, which is to describe the implementation the NLU module, and how the different elements of the module interact with the syntactic analysis expressed as a logical form.

Chapter 5
A natural language understanding module for semantic constraint extraction

This chapter provides a detailed description of the implementation of a NLU module within the grammar-based speech recognition framework described in chapter 4. The task of the NLU module is to extract a domain-specific interpretation represented as a set of *semantic constraints* for each natural language query. In this framework, constraint extraction can be seen as a mapping process where linguistic terms in the query are translated into domain-specific concepts and relations, as described in chapter 3. The task is divided into several sub-tasks, which are handled by different constituents of the NLU module. The interaction of the constituents is given in figure 11.

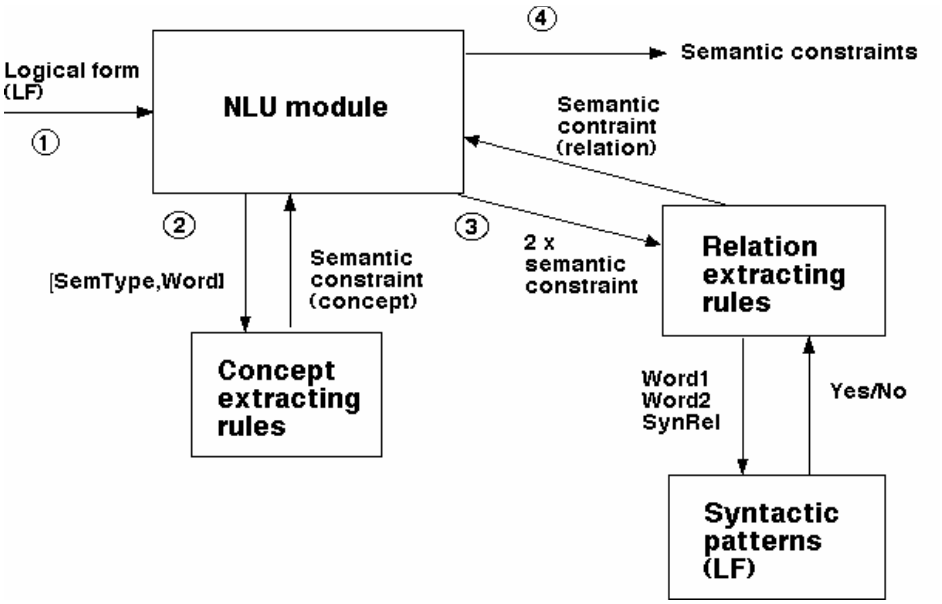

Figure 11: NLU module architecture

The input to the NLU module is a logical form representing the natural language query. The mappings between the linguistic terms in the logical form and concepts in the domain model are done with mapping-rules, which operate at two different processing levels. At the first level, the goal is to process single words or compounds. Such words or compounds can either trigger the creation of concept instances (e.g. the word *"who"* triggers the creation of an instance of the concept *person*) or provide values for the

attributes characterizing the created instances (e.g. the word *"Susan"* portions the attribute *firstname* of a person instance to the value *'Susan'*). At the second level, domain-specific relations between the identified concepts are extracted. The rules operating at this second level are triggered by pairs of concepts and use the syntactic analysis of the query as an important source of information to check whether the triggered relations are valid. The next two sub-sections describe the details of the two processing levels.

5.1 Extracting concepts

During the first phase of the semantic constraint extraction, only the lexicalized semantic information present in the logical form is treated. The lexicalized semantic information consists of a word and its semantic type. For instance, the logical form produced for "*Who suggested the coffee machine?"* has the following elements (the lexicalized semantic information is marked in bold):

(20) [utterancetype,whq,V1]
[subj,V1,V2]
[spec,pro,V2]
[human,who,V2]
[tense,past,V1]
[propose,suggest,V1]
[obj,V1,V3]
[spec,the_sing,V3]
[artefact,coffee_machine,V3]

Formally, a constraint extracting rule consists of three parts, where the second is optional:

(21) Input: [SemType, Word]
Condition on SemType
Output: sc(C,ID,A,Word)

The first part of the rule defines the input word and its semantic type, the second part specifies the possible values of the semantic type, and the last part is the semantic constraint produced by the rule. The four variables of the semantic constraint are: the name of the concept (C), a unique instance identifier that represents the instance of the concept (ID), a name of an attribute (A) and finally the word in the natural language query that is mapped to this semantic constraint. For instance, the mapping rule for the domain concept *argumentative segment* is:

(22) Input: [SemType,W]
Condition: SemType = discuss | propose | reject | accept | ask | answer | …
Output: sc(argseg,ID,class,W)

This rule applies to the element [propose,suggest,V1] in example (20), and produces the semantic constraint sc(argseg,ID,class,suggest), where ID is a new identifier (number) generated by the system.

Many mapping-rules only have two parts: the input and the output. This happens when the input word can have only one specific value for the semantic type. For instance:

(23) Input: [firstname,W]
Output: sc(person,ID1,firstname,W)

(24) Input: [familyname,W]
Output: sc(person,ID2,familyname,W)

As previously said, some words in a user query can refer to concepts in the domain without pointing to a specific attribute, for instance "*who"*, "*people"* and "*participant"*, which all refer to the domain concept *person*. Such words are processed by concept instantiation rules and produce *concept instances* (not semantic constraints). Concept instances have three variables: a concept name, an instance identifier and the word that triggered the instantiation. An example of this type of rule is:

(25) Input: [SemType,W]
Condition: SemType = human | person(sing) | person(plur) | agent
Output: ci(person,ID,W)

Concept instances don't provide any information about the attributes of an instance of a concept, but they can play a role in the second processing level, where relations between concepts are extracted. An example of this will be shown in the next section.

There is one domain concept in Archivus that cannot be processed with rules of the type described above, namely the *summary* concept, which represents the topical items of thematic episodes, argumentative segments, utterances and documents. For instance, in the query *What decision was made about the sofa?* the word *"sofa"* should be mapped to the summary item *'sofa'*. The fact that this is a summary item is not decidable from the semantic type of the word *"sofa"*. In the data model, any word can be a summary item if the word has occurred in a meeting, either in the discussion or in a document used during the discussion.

Instead, summary items can be recognized in queries through the context in which they appear. Constructions typical for such contexts are: *"about X"*, *"concerning X"*, *"discuss* X", *"propose X"* etc. where *X* is the summary item. The rules that extract summary items therefore need to access fragments of the logical form larger than just a lexicalized semantic item. Currently we have two general rules for extracting summary items, one for the prepositional constructions *"about X"*, *"regarding X"* etc. and one for the verb-object constructions *"discuss X"* etc.:

(26) Input: LF
Condition: [reference,V1,V2] [SemType,W,V2] in LF
Output: sc(summary,ID,item,W)

(27) Input: LF
Condition: 1. [SemType1,Word1,V1] [obj,V1,V2]
[SemType2,Word2,V2] in LF
2. SemType1= discuss | propose | utter | …
Output: sc(summary,ID,item,W2)

These two rules describe the most typical syntactic constructs appearing in queries with references to summary items, but there are cases where they don't apply. Queries, for which it is difficult to write appropriate rules, are discussed in section 6.3, when presenting experimental results.

5.2 Extracting relations

The second processing level in the NLU module is the extraction of relations between the identified domain concepts. The relations are defined by the data model and there are two important reasons why they should be extracted from the queries.

First of all, there are domain concepts in the data model that have relations with many different concepts. Depending on which relation is referenced in the query, the instance of the concept has different *roles*. For instance, the concept *person* has three explicit relations, and one implicit relation, with other concepts in the data model:

(28) Utterance/Person: speaker
Meeting/Person: participant
Document/Person: author
Argumentative Segment/Person: speaker

The system needs to know which of these relations the query makes reference to, in order to be able to search for the right information in the database of meeting records. Examples of three queries that refer to three different *person*-relations are shown below.

Speaker
(29) In which meeting did *Agnes present* the article about the Google culture?
Person Utterance

Author
(30) *Who* wrote the *article* about the Google culture?
Person Document

Participant
(31) Did *Susan* attend the *meeting* about the Google culture article?
Person Meeting

The second reason why it is important to extract relations is that a query can make reference to more than one instance of the same concept, and the instances may have different relations with the other concepts in the query. An illustrative example is the query *"Who rejected Agnes proposal about the coffee machine?"*. The query contains two references to instances of the concept *person* (*"who"* and *"Agnes"*) and two to the

concept *argumentative segment* (*"rejected"* and *"proposal"*). Extracting relations is, in this case, essential for deciding which person is the speaker in which argumentative segment.

We have now motivated why relations need to be extracted in a complex domain such as the meeting domain used in Archivus. In practice, the extraction is done through a set of mapping-rules, which operate in a similar way as the mapping-rules for extracting concepts. The difference is that the rules use syntactic rather than lexicalized semantic information to make the mapping between the linguistic and domain-specific representations of the query. Formally, a *relation extracting rule* has the following form:

(32) Input: sc(Concept1,ID1,Attr1,Word1) sc(Concept2,ID2,Attr2,Word2)
Condition: syntactic_relation(Word1,R,Word2)
Output : sc(Concept1,ID1,Rel,ID2)

A relation extracting rule is triggered by pairs of semantic items that were extracted during the first level of processing. Each rule specifies the names of two concepts, for instance *person* and *document*. It also specifies if the input semantic items are *semantic constraints* or *concept instances*. The output is one or several semantic constraints that describe the domain-specific relation between the two input concepts. The condition, which has to hold for the rule to apply, is that the two words, which are in the two input semantic items, participate to a specific syntactic relation in the logical form. The syntactic relations are seen as possible references to the domain-specific relations, and there may, in fact, be several syntactic relations that embody the same domain-specific relation.

The syntactic relations are specified in terms of syntactic roles (subject, object, reference etc.), which is a convenient level of abstraction to express how syntactic constructs in natural language are mapped to domain-specific relations in the data model. We give an example of a rule for the two concepts *argumentative segment* and *person* that accepts four different syntactic relations as the reference to the domain-specific relation *speaker*. Notice that the relation is *implicit* in the data model and has to be expressed with two constraints, saying that the person is the speaker of an *utterance*, which in turn occurs within the argumentative segment.

(33) Input: sc(argseg,ID1,class,Word1) sc(person,ID2,Attr,Word2)
Condition: syntactic_relation(Word2,*possessor_of*,Word1) |
syntactic_relation(Word2,*subj_relobj*,Word1) |
syntactic_relation(Word2,*subj_action*,Word1) |
syntactic_relation(Word2,*subj_obj*,Word1)
Output: sc(argseg,ID1,utterance,ID3)
sc(utterance,ID3,speaker,ID2)

The four syntactic relations are illustrated in figure 12 with four different queries, where *"Susan"* triggers an semantic constraint describing a *person* and *"decision"* an *argumentative segment*.

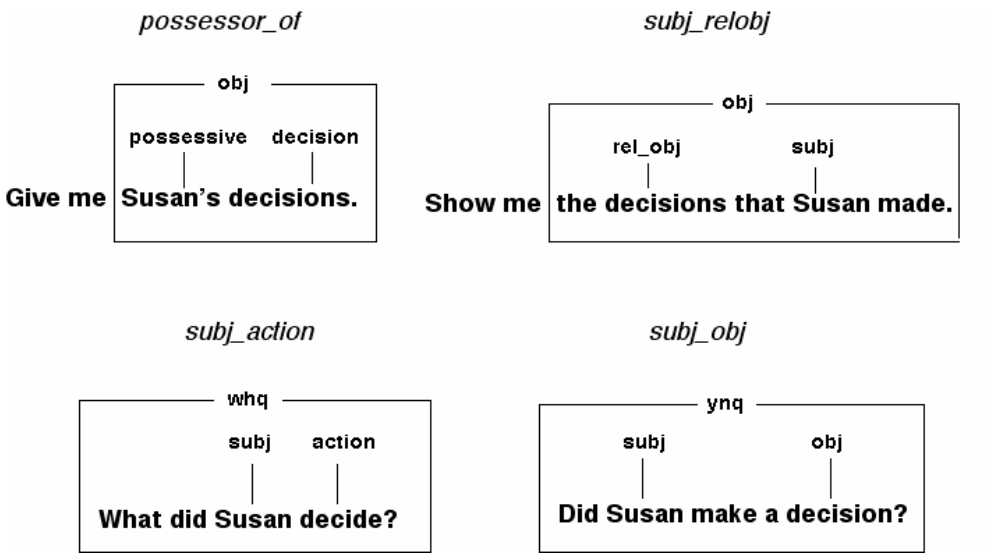

Figure 12: Syntactic relations

Notice that the above rule is not completely well-defined with respect to the syntactic relations *subject-object* and *subject-relative object*. The main verb is not taken into consideration, which means that the domain-specific relation is extracted for any verb standing between the subject and the object. For instance, "*Did Susan make a proposal?"* and "*Did Susan reject the proposal?"* are both queries where *"Susan"* is subject and *"proposal"* is object. According to the relation extracting rule, Susan is the *speaker* of the proposal in both queries, which is incorrect. Another example where the verb is important is *"Did Susan write an article?"* and *"Did Susan present an article?"* where the verb decides if Susan is the *speaker* of an utterance or the *author* of a document. To incorporate relevant information about the verb in the subject-object relations, we add a variable to these relations, specifying the semantic type of the verb. For instance, in the rule we gave as an example, we modify the subj-obj and subj-relobj conditions to:

(34) VerbSem = utter | inform | create | activity | provide
syntactic_relation(Word1,subj_obj,Word2,VerbSem)
syntactic_relation(Word1,subj_obj,Word2,VerbSem)

With the restriction imposed on the semantic type of the verb, the rule applies for subject-object relations where the verb is: *make, have, present, express, provide, give etc.* but not where is it for instance *write, accept* or *reject*.

A syntactic relation is checked by searching for a specific syntactic pattern between the two words involved. What the concrete nature of the pattern depends on the type of syntactic analysis available (logical form or compact tree forest). In a logical form, the syntactic roles of the words are provided in an explicit way, which makes it very easy to specify the precise pattern that corresponds to a syntactic relation. For instance, the possessor_of relation can be expressed with the following pattern:

(35) syntactic_relation(Word1,possessor_of,Word2) : [possessive,V1,V2]
[SemType1,Word1,V2]
[SemType2,Word2,V1]

This pattern occurs in *"What was Susan's decision?"* where *"Susan"* is the possessor of *"decision"*. The logical form for this query is illustrated below and the parts corresponding to the syntactic pattern of the possessor_of relation are marked in bold:

(36) [utterancetype,whq,v1]
[subj,v1,v2]
[spec,pro,v2]
[pronoun,what,v2]
[tense,past,v1]
[exist,be,v1]
[obj,v1,v3]
[possessive,v3,v4]
[spec,pro,v4]
[firstname,susan,v4]
[decide,decision,v3]

For subject-object relations, where the semantic type of the verb needs to be specified, the pattern is expressed as:

(37) syntactic_relation(Word1,subj_obj,Word2,**VerbType**) :
[subj,V1,V2]
[SemType,Word1,V2]
[obj,V1,V3]
[SemType,Word2,V3]
[**VerbType**,Verb,V1]

The advantage of encoding the syntactic patterns into separate rules (or "functions") instead of specifying them directly in the relation related rules is that it makes the processing of the queries very modular. If the syntactic representation of the query is changed, for instance to standard syntactic derivation trees, the relational rules do not have to be adapted. Only the specifications of the syntactic patterns need to be updated. We will discuss in section 7.1 how the NLU module can be plugged into a different natural language processing architecture, where the syntactic analyzer produces a forest of syntactic trees, instead of one (or several) logical form(s).

Chapter 6
Validation

6.1 Implementation status

A prototype of the NLU module has been implemented in Prolog according to the description in chapter 5. The rules for extracting instances of concepts and relations between pairs of concepts are all based on a corpus of 80 natural language queries that represent the type of queries that users are foreseen to ask about meetings. The queries have been gathered by several people involved in different parts of the Archivus development.

The current implementation of the prototype has 25 concept related rules and 30 relation related rules. In total 11 different syntactic relations are used by the relation related rules. Some of them, not mentioned in the previous examples, are *subject-action*, *action-object*, *reference*, *of*, *to* and *from_location*. The reason why they are relatively few is that the same syntactic relations are used by many different rules, for different combinations of concepts.

The lexicon used by the speech recognizer is still in a development stage, which means that the semantic types described in section 4.4 are not yet fully implemented in the lexicon and were not available at the time of testing. To be able to do some experiments, another ad-hoc semantic lexicon was created in Prolog, which simply lists all the words in the corpus of 80 queries grouped into different semantic classes, some of which are purely domain-specific ones: *first name*, *family name*, *meeting*, *document*, *person*, *institute*, *argumentative class*, *create*, *communicate*, *provide*, etc.

Also the grammar is not yet completed, which means that some of the queries in the corpus receive no logical form, and therefore cannot be processed by the NLU module. Variations of such queries compatible with the current version of the grammar were tested instead.

6.2 Evaluation methodology

The goal of the preliminary experiments is to verify whether the concept and relation related mapping-rules are well-formed and whether the syntactic relations are adequate for controlling the extraction of domain-specific relations from the queries. The 80 queries in the corpus are parsed with the available version of the grammar, and one of the analyses (there can be several) is then processed by the NLU module. Speech recognition errors are not considered at this stage. Only written queries are tested. As

soon as the grammar and lexicon are finished, they will be used to build an adequate grammar-based speech recognizer, and vocal queries will also be tested.

The semantic constraints have been extracted manually for all the queries in the corpus and serve as the reference. In the evaluation of the automatically extracted constraints, the queries are divided into four groups that represent how closely the extracted constraints correspond to the reference ones. The four groups are:

1. Full extraction: All the semantic constraints were extracted correctly
2. Partial extraction: Some of the semantic constraints in the reference were not extracted automatically
3. Incorrect extraction: Some of the semantic constraints that were extracted automatically were not found in the reference
4. No extraction: The NLU module did not extract any semantic constraints from the query

For the queries that have no result, a distinction is made between the cases where the syntactic analyzer failed to produce a logical form, and where the NLU module failed to extract concepts.

6.3 Experimental results

In this section we give the experimental results of the extraction of semantic constraints from 80 queries, as described in the previous section. The results are presented in table 1. Among the 80 queries that were tested, there was no case of an incorrect extraction, or cases where the NLU module would fail to extract something from a syntactic analysis that was provided. Nevertheless, almost half of the queries in the test corpus had no syntactic analysis. This is due to the grammar still being in the development stage, but in many cases, the parse failure could be overcome by changing a single word in the query. To be able to test the semantic constraint extractor on these queries, we made the necessary modification to the queries to obtain the syntactic analyses. Typical modifications were changing a preposition for another (e.g. *"on"* to *"of"*), or a proper noun for a pronoun (e.g. *"Susan's opinion"* to *"her opinion"*). These changes were considered to have no important effect on the semantic constraint extraction, as the logical form would be almost identical and the number of extractions would remain the same. A third type of change was to remove a prepositional phrase, e.g. by changing "*Who suggested the solution to solve the problem with the white board*?" to "*Who suggested the solution to solve the problem?"* This type of change was applied in only 5 cases, because it simplifies the query too much by reducing the number of semantic constraints that the NLU module needed to extract.

From the 32 queries that had no syntactic analysis, altogether 21 would be modified so that a logical form was obtained and the query processed with the NLU module. The remaining 11 queries were left unchanged because they were too far from what the grammar was able to analyze.

	Full extraction	Partial extraction	No analysis
Original 80 query set	36%	24%	40%
Modified 80 query set	59%	28%	14%

Table 1: Experimental results

When studying the queries that were *successfully* processed by the NLU module, it is important to observe that there are quite substantial differences in the level of complexity. The most simple ones (28%) are those that produce a single semantic constraint, e.g.

(38) *What decisions were made?* sc(argseg,1,class,decision)

Then we find queries of middle complexity that have a mapping to two concepts and one relation (38%), giving a total of three semantic constraints, e.g.

(39) *Who participated to the meeting?* ci(person,1,who)
ci(meeting,2,meeting)
sc(meeting,2,participant,1)

The most complex queries involve at least three concepts and two relations. 34% of the successfully processed queries belonged to this type. Among the queries, for which a partial extraction succeeded, the distribution was 47% middle and 53% complex. The numbers indicate that it is not the *complexity* of the query that induces the main problems in the extraction. To find out what other factors are involved, we looked more closely at the queries that received a partial analysis. Several common features were found.

Going back to the implementation of the extraction rules, a strong assumption has been made about the mapping between syntactic relations in the query and domain-specific relations in the data-model. The assumption is that a relation is encoded by two words (representing two concepts) that are interconnected through a syntactic pattern, e.g. subject-action. But in some queries encountered in the test corpus, single words can refer to both a concept and a relationship. For instance, the word *participant* refers to the concept Person and the relation *participant* between Meeting and Person. A query such as "*Who were the participants?"* should give the following semantic constraints:

(40) ci(person,1,who)
sc(meeting,2,participant,1)

Instead the extraction rules produce:

(41) ci(person,1,who)
ci(person,2,participant)

Since there is no word in the query referring explicitly to the concept *meeting*, the participant-relation cannot be extracted by the rules. Additional special rules need to be considered to cover this phenomenon.

A second problem is the extraction of *summary items* and the different relations they have with other concepts in the query. In the current implementation, there are two rules that identify summary items based on simple syntactic patterns: verb-object (e.g.*discuss X*) and reference (e.g. *about X*). These rules are quite sufficient if the summary item is a single word or a short noun phrase. But the rules are not able to handle cases such as *"Was someone opposed to having a computer?"* where the syntactic pattern is *opposed to X,* and *X* is a longer phrase containing the summary item *computer*. Some more sophisticated rules are required also for this type of queries.

Even when a summary item is successfully identified, it is not always the case that the relation between the summary item and another concept in the query is extracted properly. The relations are extracted with the same syntactic patterns as the summary items, i.e. that A *references* B or that A is in *verb-object* relation with B. When there is ambiguity in the syntactic analysis, one of the analyses is chosen arbitrarily as the input for the NLU module. Sometimes the "wrong" one is chosen, which means that the syntactic relation checked by the NLU module does not exist in the logical form, although it exists in one of the other solutions. For instance, the query *"What was the outcome of the discussion about the sofa?"* is ambiguous with respect to the scope of the prepositional phrase "*about the sofa"*. Two segmentations are possible:

(42) [What was the outcome [of the discussion [about the sofa]]]
(43) [What was the outcome [of the discussion] [about the sofa]]

To illustrate the problem, we describe the concrete processing of the query. In the first level, the NLU module extracts two concepts from the logical form, regardless of which logical form is chosen: an Argumentative segment (discussion) and a Summary (sofa). On the next level, a relational rule attempts to extract the relation *argumentative segment summary* by checking if the appropriate syntactic relation exists between *discussion* and *sofa*, in this case *"discussion references sofa"*. In analysis (41) this relation exists, in (42) it does not. So the choice of syntactic analysis has an impact on the success of extracting conceptual relations, and some care has to be taken in selecting the right syntactic analysis. Alternatively all analyses (if not too many) might be processed and potential conflicts/inconveniences then resolved.

Chapter 7
Discussion

In the previous chapters we described the implementation of a NLU module that extracts semantic constraint from queries using lexical and syntactic mapping rules. We also provided some preliminary tests on a corpus of queries within the Archivus domain. Two main issues need to be discussed. The first is how the architecture of the NLU module can be plugged into a more robust environment, where the syntactic analysis is provided as a compact forest of derivation trees instead of a single logical form. The second issue is how the semantic constraints produced by the NLU module need to be processed further to be useful for the Archivus dialogue manager. Therefore we will address some of the main problems related to these issues.

7.1 Semantic constraint extraction in a robust environment

The approach to natural language understanding that is based on extracting information from a logical form has one obvious weakness, which is that the query has to be fully analyzable by the syntactic grammar in order to provide a logical form. As pointed out in section 4.1, this can have the consequence that many queries will not be parsed and will therefore not provide any input to the NLU module. The system fails to "understand" although there may have been fragments in the query that the syntactic analyzer was able to parse. If these fragments could be processed, it would enable the NLU module to extract at least some semantic constraints even if the whole sense of the query might not be captured.

The solution for making the architecture more robust is, first of all, to use a speech recognizer that instead of returning 1 or n-best transcriptions returns a word-lattice. This word-lattice is then parsed with a CFG grammar, and the resulting derivation trees are stored in a chart. How the speech recognition is tuned to provide a word-lattice of good quality, and what techniques are to be used in the parsing process, are out of the scope of this report. Only the impact of the nature of the representation of the syntactic analysis is of interest here. Assuming that the CFG derives trees the nodes of which are syntactic categories (NP,VP,etc.), the syntactic patterns that are checked by the NLU module need to be specified in terms of syntactic *categories* rather than syntactic *roles*. An important difference between the two approaches is that a syntactic relation, encoded as syntactic roles, corresponds to only one syntactic pattern in a logical form, whereas the same syntactic relation can correspond to several different syntactic patterns operating on syntactic trees. For instance, the two queries *"Which proposal was rejected by the members of the team?"* and *"Who disagreed with the solution about the chairs?"* instantiate different syntactic patterns for the subject-action relation (one being for active

construction and one for passive). To express the patterns in a flexible way, one may view the tree as a string of characters, consisting of brackets, non-terminals and terminals. For the two queries above, for instance, the strings would be:

(44)
```
[s
  [np
    [d(which),
     nbar[n(proposal]],
   vp
    [vbar[v(was)],
     vp[vbar[v(rejected)],
        pp
          [p(by),
          np
            [d(the),
             nbar[n(members)],
             of_pp[
                    of,
                    np
                      [d(the),
                       nbar[n(team)]]]]]]]]
```

(45)
```
[s
  [np[pronoun(who)],
   vp
     [vbar[v(disagreed)] ,
      pp
        [p_with_agr(with),
         np
           [d(the),
            nbar[n(solution)],
            pp
              [p(about),
               np
                 [d(the),
                  nbar[n(chairs)]]]]]]]]
```

The subject-action pattern can then be expressed as a *regular expression* (in a language especially designed for parse trees) that describes the relevant parts of the string to be matched:

(46)
```
[s [ np %* [nbar [ . "W1" ]] %*] [ vp %* [vp[vbar [v "W2"]] %*] %*] |
[s [ np %* [. "W1"] %* ] [ vp %* [vbar [v "W2" ]] %*] %*]
```

The *"%"*-symbol means "any subtree" whereas the dot *"."* means "any non-terminal". Regular expressions have the advantage that they can be instantiated with powerful finite-

state automata techniques to efficiently search for sub-trees in a potentially huge chart of derivation trees. The main difficulty with this approach is to define the right regular expressions. They have to be written manually, and in order to deal with the large variation of syntactic constructions, example queries have to be parsed and their derivation trees need to be observed one by one. An interesting question is whether these patterns can be produced with automatic techniques, using machine learning methods applied on large example corpora.

Another question that arises, when checking syntactic patterns in a chart instead of a single syntactic analysis, is how consistent the extracted constraints might be. For one query, there may be several different relations that need to be checked between different combinations of words. As the chart is a compact representation of a forest of trees that partially overlap, and if two syntactic patterns are checked independently, and both are found in the chart, it is not guaranteed that they belong to the same derivation tree, i.e. that their concatenation constitutes a consistent match in the chart. However, it is not obvious that this is a problem. The chart contains analyses for different recognitions proposed by the speech recognizer, and since the speech recognizer makes errors, it is not guaranteed that there exists a match in the chart that corresponds to the complete and correct analysis of the vocal query. Each tree can be partially correct, and by extracting information from several of them, the maximum number of correct semantic constraints can be extracted.

The main issue is then to decide how to deal with the possibly huge number of semantic constraints that the NLU module might extract. There can be alternative interpretations that are inconsistent with each other, and from this set of constraints the right interpretation has to be selected. Scoring functions are probably needed to rank the semantic constraints, but the precise nature of the scores remains to be explored.

7.2 Using the results of the semantic constraint extraction

The second issue of discussion is how to use the semantic constraints in the concrete application. In Archivus, the identified constraints are used to feed the dialogue manager that tries to fill slots in a dialogue frame, as described in section 2.4. In this perspective, the semantic constraints are not immediately usable for the dialogue manager. Indeed, they describe the elementary concepts in the data model (*person, meeting*, *argumentative segment* etc.), and, as such, do not directly map onto the attribute-value pairs used in the dialogue frame, which are much more specific (*SpeakerAdressCity*, *MeetingParticipant-AddressCity*, etc.). In fact, the attributes are a compact representations of a combination of semantic constraints, which should mean that the transformations required on the semantic constraints to provide attribute-value pairs should be simple merging and reductions. Most of the transformations are straightforward, e.g.

(47)
sc(person,1,firstname,Susan)
sc(location,2,city,Geneva)
sc(person,1,address,2)
ci(utterance,3,present)
sc(utterance,3,speaker,1)

→

SpeakerFirstName=’Susan’
SpeakerAddressCity=’Geneva’

But some problems are encountered. For instance, the dialogue manager does not consider complex queries that involve argumentation between several speakers. The slots in the dialogue frame do not link a speaker to an argumentative class, so if a query refers to more than one speaker and argumentative segment, the relations between the speaker and the argumentative segment is lost.

Another problem is how to deal with cases where semantic constraints are extracted only partially. For instance it is possible that a semantic constraint describes an instance of a person, but that no semantic constraint is provided to describe its relation to other concepts. Then it is not possible to decide if the person is a participant, speaker or author, unless additional information (for instance the dialogue state) is available that allows prioritizing one before the other. However, the question is open whether resolving such ambiguities is the task of the NLU module or of the dialogue manager.

Chapter 8
Conclusions and future work

In this report we have presented a method for extracting "meaning" out of natural language queries by mapping lexical and syntactic elements present in the query to a domain-specific representation called semantic constraint sets. The extraction relies on the fact that there is a fixed set of concepts and relations that are defined in the application domain. Only queries that reference these concepts and relations are "understood".

The model-driven approach to natural language understanding is relatively fast to prototype, because the main tasks are (1) to build a dictionary of words that covers the user vocabulary and to provide some semantic information about these words (e.g. using a conceptual resource such as WordNet or EDR) (2) to identify syntactic patterns in the natural language queries that correspond to the conceptual relations in the domain model and (3) to write rules that make the mappings on the two levels.

Because the application domain is very specific, the dictionary is not very large and the task of adding semantic information to the words is not very time-consuming. Also using the notion of *syntactic roles* to specify the syntactic relations is a good level of abstraction to express what information in the syntactic analysis is actually important for the understanding. The syntactic relations are highly reusable as the same syntactic relation applies to many different domain-specific relations.

Separating the processing levels related to concepts and relations into different modules shows that there is a clear distinction between how these two types of linguistic information (lexical and structural) contribute to the extraction of semantic constraints. Designing the system in this modular way also makes it easy to test it during the development stage by simply adding and removing rules in the different modules.

Preliminary tests show that the relation related rules are applicable in the majority of cases, but are not sufficient for some queries, where words can refer to both a concept and a relation in the domain model. A future task is to consider different ways of designing mapping rules to handle such cases, either by adding new rules in the existing modules or by introducing a new module that operates on the output from the previous ones.

The main focus of the future work is to implement the second, robust architecture of the NLP component, in which ungrammatical or badly recognized queries are handled. An interesting experiment is to compare the two NLP-paradigms, to determine if the lack of robustness in the grammar-based approach is as acute as it seems, and if the performance of the robust approach (in terms of accuracy) is competitive with the grammar-based one.

To be able to make a true evaluation of the usability of the two NLP architectures, more data needs to be collected from the users of the system. The corpus of queries that was used during the development of the linguistic resources and the NLU module reflects what the developers of the system *think* that users will say. In the near future, Wizard-of-Oz experiments will be conducted to gather non-biased user queries. These queries will be important for discovering new linguistic phenomena that were not present in the initial corpus, and to make a judgment of how realistic the previously gathered queries are. It is possible that long and complex queries are very rare in real life and do not need to be treated with sophisticated methods. The goal is to have a practical system that is able to handle the queries that occur most frequently in the interaction between the user and the system.

Bibliography

James Allen. 1995. “Natural language understanding”. *Menlo Park, CA. Benjamin/Cummings.* Second edition.

James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. 2001. "Towards Conversational Human-Computer Interaction," *AI Magazine*.

L. Androutsopoulos, G.D Ritchie, P.Tranisch. 1995. “Natural Language Interfaces to Databases - An Introduction”. *Journal of Language Engineering.*

T. H. Bui, M. Rajman and M. Melichar. 2004. “Rapid Prototyping Methodology”. *Proceedings of the 7th International Conference on Text, Speech Dialogue (TSD 2004), P. Sojka, I. Kopecek K. Pala (Eds.), Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg New York pp. 579-586*, Brno, Czech Republic, September 8-11, 2004.

R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, G. B. Varile, A. Zampolli. 1997. "Survey of the State of The Art in Human Language Technology". *Studies in Natural Language Processing, Cambridge University Press, 513 pages*

C. Fellbaum (ed.). 1998. “WordNet: An electronic lexical database”. *MIT Press.*

S. Harabagiu, M. Pasca, and S. Maiorano. 2000. “Experiments with open-domain textual question answering”. *In Proceedings of COLING-2000*, Saarbruken Germany, 2000.

S. Knight, G. Gorrell, M. Rayner, D. Milward, R. Koeling and I. Lewin. 2001. “Comparing grammar-based and robust approaches to speech understanding: a case study”. *In Eurospeech 2001.*

S. Larsson, P. Ljunglöf, R. Cooper, E. Engdahl and S. Ericsson. 2000. “GoDiS - An Accommodating Dialogue System”. *In Proceedings of ANLP/NAACL-2000 Workshop on Conversational System.* Seattle, USA, May 2000.

A. Lisowska M. Rajman, M. T.H. Bui. 2004. “ARCHIVUS: A System for Accessing the Content of Recorded Multimodal Meetings”. *MLMI’04 Workshop on Multimodal Interaction and Related Machine Learning Algorithms.*

V. Pallotta, H. Ghorbel. 2003. “Argumentative segmentation and annotation guide-lines”. *Technical report IM2.MDM-08.*

M. Rayner. 1993. “Abductive Equivalential Translation and its Application to Natural Language Database Interfacing”. *PhD thesis*, Stockholm University/Royal College of Technology.

A.I. Rudnicky, E. Thayer, E., P. Constantinides, C. Tchou, R. Shern, K. Lenzo, W. Xu, A. Oh. 1999. “Creating natural language dialogs in the Carnegie Mellon Communicator system”.*Proceedings of Eurospeech, 1999, 4, 1531-1534.*

N. Stratica, L. Kosseim and B.C. Desai. 2003. “NLIDB Templates for Semantic Parsing”. *Proceedings of Applications of Natural Language to Data Bases (NLDB 2003). pp. 235-241,* June 2003. Burg, Germany.

W. Wahlster. 2000. “Pervasive Speech and Language Technology”. *In: Wilhelm, R. (Ed.): Informatics - 10 Years Back, 10 Years Ahead. Berlin, Heidelberg, New York: Springer, Lecture Notes in Computer Science Vol. 2000, pp. 274- 293*.

Z. Zheng. 2002. “AnswerBus question answering system”. *In Human Language Technology Conference (HLT).*

T. Yokoi. 1995. “The EDR electronic dictionary”. *Communications of the ACM, volume 38, issue 11, pages 42-44. ACM Press.*

Regulus: http://sourceforge.net/projects/regulus