Using Application-Specific Ontologies to Improve Performance in a Bottom-up Parser

Marta Gatius **Meritxell González**
TALP Research Center
Technical University of Catalonia
Barcelona, Spain
{gatius,mgonzalez}@lsi.upc.edu

Abstract

This paper describes how the application specific knowledge represented in ontologies is used to improve the processing of user interventions in a multilingual dialogue system for multiple applications. The user interventions are processed by a left-corner parser performing syntactic and semantic analysis in parallel. The parser uses application-restricted grammars and lexicons obtained from ontologies representing the application specific knowledge. It also uses the knowledge of the dialogue context to select the grammar rules related to the dialogue focus.

1 Introduction

Many works have focused on the study of approaches that improve parsing in real-world applications. Several of these approaches propose new mechanisms to practical parsing speed-ups (Boullier, 2003), while others propose the use of semantic information to improve the parsing results (Mohanty and Balabantaray, 2003). This paper describes how the performance of a left-corner parser used in a practical dialogue system is improved by using knowledge from domain ontologies and context dialogue.

The Principle of Compositionality states that the meaning of a sentence can be composed of the meaning of its syntactic constituents. There are two different approaches following this principle: a pipeline architecture where the result of the parsing is passed as input to the semantic analyzer, and the integration of semantics directly into the parser. In this second approach, when performing syntactic and semantic analysis in parallel, the representation of the meaning for a constituent is created once all its constituent parts have been recognized. The main advantage of this approach is that semantic considerations can be used to reduce the number of possible syntactic structures to be considered.

However, the integration of semantic and syntactic analysis has also disadvantages. The primary of them being that considerable effort is spent on the semantic analysis of constituents that, in the end, do not contribute to a successful parse. Additionally, the traditional wide-coverage grammars are not particularly well-suited for compositional semantic analysis because they include many general syntactic constituents with no semantic role. For these reasons, many practical systems requiring semantic analysis for a particular domain use grammars adapted to the specific domain. As explained in (Jurafsky and Martin, 2000), these grammars simplify semantic processing in several ways:

- Their rules and constituents are designed to correspond directly to entities and relations from the domain.
- The key semantic components occur together within single rules while in traditional grammars the key semantic elements are often widely distributed across parse trees.
- Their nature enables a certain amount of prediction about upcoming input, thus helping with anaphor and ellipsis phenomena.

However, domain-restricted grammars and lexicon present several limitations. The possible sentences that can be analyzed correctly with domain-restricted resources are limited. Besides, the number of grammar rules needed for a particular domain can be high because syntactic generalizations are limited. Furthermore, application-restricted resources are expensive to develop and difficult to reuse.

This paper describes the approach followed when applying syntax-driven semantics to a practical dialogue system for multiple applications. This system supports speech and textual

interaction in several languages (English, Spanish, Catalan and Italian). The voice components of the system are based on VoiceXML. The automatic speech recognition system uses grammars to recognize and process user interventions. These grammars incorporate semantic information (following the standard Speech Recognition Grammar Specification). The text input is processed by a chart-based parser, implemented in Prolog, with top-down filtering that performs the syntactic and semantic analyses in parallel. A more detailed description of it can be found in (Gatius, 2001).

In this dialogue system the application knowledge involved in the communication is represented in ontologies. These ontologies are used to generate the dialogues, the system messages and the linguistic structures necessary for processing user interventions. In this paper we focus on how the application specific knowledge represented in ontologies is used to improve the performance of the parser.

In the current implementation of the dialogue system the parser is only used to process text input. Basically, it is used to study how the voice module could be improved. For this reason, the grammars and lexicons used by the parser to process textual input are very similar to those used by the speech recognizer. For the next version of the system we are planning to integrate the parser with the voice module, using the last VoiceXML version.

As to the structure of this paper it is as follows: in the following section we describe the use of ontologies modelling the application tasks-specific knowledge for dialogue management. Section 3 describes how the application information represented in the ontologies is associated with the linguistic resources and how this information is used by the parser to interpret the user's interventions. Section 4 explains how the contextual information is used to improve the efficiency of the parser. Results of the parser performance are discussed in Section 5. And finally, the last section draws some conclusions.

2 Ontologies and Dialogue Management

Dialogue management techniques deal with the use of dialogue in furthering the user's objectives. Dialogue management is related to the type of interaction allowed, as well as the underlying task and domain.

This section describes briefly the different approaches to dialogue management according to how task models and dialogue models are used. Then it describes the approach followed by the dialogue system presented in the introduction, that is, the use of ontologies for modelling the application tasks. The following sections describe how these ontologies are used to improve the processing of user's interventions.

2.1 Dialogue Management Models

Almost all dialogue systems use a dialogue management model defining the dialogue control and guidance, although in many of them this model is not explicit. Dialogue management models have evolved from simple finite state automata, representing the questions asked by the system and all the actions to take depending on the user's response, to more complex models using dialogue and tasks models to provide generic behaviour.

The most simple dialogue management models are finite state models. They are still used in many spoken systems. In these models both dialogue model and task model are implicit. They are appropriate for simple and structured tasks but not for complex applications, when the number of possible interaction paths grows.

Many spoken question-answering systems use a more flexible dialogue management model, based on frames (or templates) representing the task model. These frames contain slots describing the various kinds of information the user would be asked to fill. The number of frames needed depends on the application tasks complexity. In most of those systems there is no explicit dialogue model. Frame-based system models are appropriate when the set of actions the system can do is reduced (being the number of frames limited), which is not the case in complex applications. The main problem in finite state and frame-based models is that they only support very limited user initiative.

In more complex systems a more flexible communication is provided by including explicit dialogue model as well as task model. Several dialogue systems supporting flexible communication in complex domain also include an explicit representation of the domain model. An example of a system using a dialogue model, a task model and a domain model is described in [9]. In this system, the intentional structure underlying the dialogue is derived from an ontology modelling the application tasks and the informational structure is obtained from the domain ontologies.

One of the advantages of using ontologies for representing the application tasks in comparison

to other conceptual representations, such as frames, is that in ontologies relations and preconditions between different entities are defined. Therefore, ontologies allow more flexible dialogues. The use of ontologies has proved especially appropriate for dialogue systems supportting different languages and modes (Wahlster, 2003). In those systems the semantic interpretation of the user's intervention is represented in basis of the same ontology (independently of the mode and the language).

An additional benefit of using ontologies is that they facilitate the reuse of knowledge common to different applications, such as tasks obtaining personal data (address, telephone number, etc.).

2.2 Ontologies for Modelling the Application Task-specific Knowledge

The dialogue system we present uses ontologies representing the tasks models. This system (as most existing dialogue system) is driven by specific goals or tasks: finding a particular piece of information (informational dialogues) or executing a particular transaction (transactional dialogues). These tasks can be decomposed into hierarchically ordered subtasks. We proposed the representation of these tasks in ontologies improves the communication.

The dialogue system supports task-oriented dialogues. In these dialogues the application-dependent knowledge appearing in communication is related basically to the tasks the application can perform, that is, the information required to perform the task and the information resulting from it. Mainly, this information is related to the application parameters. This information can be represented by an ontology. In this ontology, all application tasks are represented by a set of concepts described by attributes. Basically, these attributes describe the input and output parameters. Preconditions governing in which context a particular attribute value must be asked to the user can also be included.

The dialogue manager uses this ontology to control both the interactions with the user and the access to the application back-end. The system asks the user to introduce the data required by the application at each state of the communication. Dialogue interactions consist mainly in asking and giving the values of the conceptual attributes describing the application tasks.

Using the ontologies describing the application-specific tasks the system supports different types of dialogues. It will drive the interaction asking inexperienced users all the information the application needs (represented as attributes describing the application tasks). Experience users can take the initiative and give specific information. Then, considering context dialogue and preconditions in the concepts descriptions, the dialogue manager will decide which specific data is needed from the user. If additional data is required, the system will ask for it. If a user's answer gives information not only about the attribute she is asked for but also about other attributes, this information will be kept and will not be asked in future interventions.

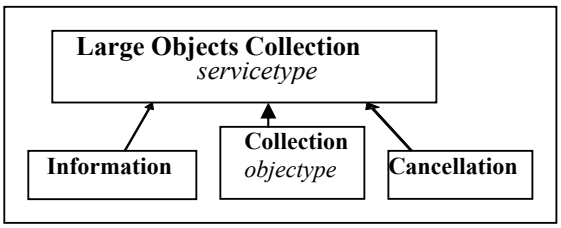

Figure 1. A fragment of the application Large Objects Collection

Figure 1 shows a fragment of the ontology representing the application to set a date to collect objects from homes. It is a transactional application performing three tasks: gives information, sets a date for collection and cancels collections previously arranged. Following the application description in Figure 1 the dialogue system will first ask the user the particular task she/he is interested in (i.e., *Say what you want, information, collection or cancellation*). If the user wants to set a data for collection, the dialogue system will then ask the type of object that has to be collected.

In complex and changing domains further improvements in communication can be achieved by using ontologies describing the domain-specific knowledge. Ontologies describing domain concepts and relations can be used for reasoning. Simple inferences, whether a class can be related to another by the *isa*-relation or other relations, such as part-whole, can help the system to detect differences in expectation in the user's interventions, such as under/over specification (corresponding to hyperonym or hyponym). In current implementation domain, ontologies are only used to obtain all possible values of specific concept attributes, as explained in next section.

3 Using Ontologies for processing user's interventions

The advantages of using ontologies to relate the application specific knowledge to the linguistic resources needed in communication in multiple domain dialogue systems have been studied in several works (Dzikovska et al., 2003). This section describes the approach we follow to associate the information in the application-specific ontologies with the corresponding linguistic resources and how it is used by the parser to interpret the user's interventions.

3.1 The semantic analysis

The semantic analysis is based on *lambda calculus*. Semantic information is associated with each grammar rule to indicate the order of interpretation of its constituents. This information consists of a list of numbers representing the constituents. Figure 2 shows examples of application-restricted grammar rules together with their semantic interpretation. This semantic interpretation associated with each lexical entry consists of a lambda function or a lambda value. Figure 3 shows examples of lexical entries and their interpretation.

At run-time, once the parser has recognized all constituents of the rule, they are analyzed semantically. The semantic analysis consists in applying the semantic interpretation associated with the categories (the *lambda functions* over the *lambda values*) following the order indicated in the semantic list associated with the rule. The resulting semantic information is incorporated into the state stored in the chart. The semantic interpretation process returns a list of possible interpretations that will be passed to the dialogue component.

The semantic interpretation built by the parser will subsequently be passed to the dialogue manager. By using the application-restricted grammar and lexicon the parser obtains the semantic interpretations in the form needed by the dialogue manager. Each possible semantic interpretation is a list of words containing the identifiers of the ontology concepts, their attributes and values. Basically, this information represents the application operations, their parameters and values. Finally, the dialogue manager will complete the information in order to perform the corresponding actions (asking the user or accessing the application back-end).

3.2 The application-specific information associated with linguistic resources

The linguistic resources for each language consist of a general grammar and a general lexicon to process expressions common to all applications and application-specific grammars and lexicon. The application-restricted linguistic resources cover the expression of the entities in the ontologies representing the application tasks. The lexical entries expressing the concepts, attributes and their values are associated with their corresponding identifiers. In order to facilitate reusability, the linguistic resources expressing parameters appearing in more than one application (i.e., dates and addresses) are incorporated into the general grammar and general lexicon.

Examples of the linguistic resources obtained from the application description in Figure 1 are shown in Figure 2, Figure 3 and Figure 4. Figure 2 shows examples of the grammar rules used to process the user's answers when the dialogue system asks the two attributes (*servicetype* and *objectype*). The lexical entries appearing in those rules are shown in Figure 3 and Figure 4.

```
gramservicetype  -> ncollection
gramobjectype  -> nobjection
gramobjectype -> ncollection prepof ngobject  (2(1 3))
ngobject -> number objectype  (1 2)
ngobject -> objectype
```

Figure 2. Examples of grammar rules

Category	Interpretation	String
ncollection	(servicetype, collection)	collection
prepof	(((l,X),(l,Y)),(X,Y))	of

Figure 3. Examples of lexical entries

The linguistic resources built for the first prototype of the system are limited. We had decided to use grammars and lexicon that could automatically be represented in the voice grammars (following the standard Speech Recognition Grammar Specification). For this purpose, the information associated with the lexical entries is very simple.

We considered the key semantic information is that provided by the expression of the ontology concepts, their attributes and values. For this reason, the semantic representation associated with a lexical entry expressing these entities is a lambda value. In the case of concepts and attributes lambda values are represented by the identifiers of the corresponding concepts or attributes. The lambda value associated with the lexical entries expressing the value of conceptual attributes

is represented by a list containing the attribute identifier and the value identifier. This information can be obtained automatically from the conceptual attributes in the ontology. The lexical entries not expressing relevant information for the application are associated with a lambda function over one or more lambda values.

Figure 3 shows two examples of lexical entries together with their semantic interpretation. The two lexical entries consist of three fields: category, semantic interpretation and string (or linguistic realization). The first lexical entry corresponds to the noun *collection*, expressing the service *collection*. The interpretation associated with this noun is a lambda value obtained from the ontology. This interpretation indicates that the noun *collection* expresses that the value of the attribute *servicetype* is the concept that has as identifier *collection*.

The second lexical entry in Figure 3 corresponds to the preposition *of*. This entry belongs to the general lexicon. The semantic interpretation associated with this preposition is a lambda function represented by the list *(((l,X),(l,Y)), (X,Y))*. The first sublist indicates that the function has two lambda arguments, represented by the variables *X* and *Y*. The second sublist establishes that the function returns a list containing the value of the two variables. This semantic interpretation indicates that the preposition *of* acts as a link between two semantic values.

The dynamic entries

Category	Dynamic function
number	integer
objectype	member(objectype)

Figure 4. Examples of dynamic entries

In order to improve the efficiency and friendliness of the communication we have incorporated dynamic categories. Their superficial representation, as well as the semantic interpretation associated with them, is set at run-time. The use of dynamic entries reduces the number of lexical entries to be considered and allows the user to introduce new values. Dynamic entries consist of two fields: the linguistic category and the dynamic function. At run-time the dynamic function processes the word(s) introduced by the user and returns the corresponding semantic information. There are two different types of dynamic entries: those representing open values (such as numbers, dates and telephone numbers) and those associated with a list of values (i.e., locations).

The first lexical entry in Figure 4 is an example of dynamic entry representing an open value. This lexical entry represents a number and is associated with the prolog predicate *integer*. The number expressed by the user would be associated with a lambda value represented by the word *number* and the numeric value. Entries representing open values can be associated with specific functions controlling the correctness of the values introduced by the user (i.e., dates, telephone numbers).

The second lexical entry in Figure 4 is an example of entry associated with a list of values. The main advantage of this way of representing the entries is that the list of values can easily be changed when there is a change in the application. It can be changed even at run-time. Another advantage of using entries associated with a list is that if the number of values is reduced those values can be presented as a menu at run-time. Then, the user has to choose only one or more values in the menu.

The dynamic entry *objectype* represents the object that has to be collected, that is, the possible values of the attribute *objectype* of the concept *Collection* in Figure 1. The function associated with the entry controls that the value introduced by the user corresponds to an element in the list *objectype*, containing the linguistic realization of the possible types of objects that could be collected.

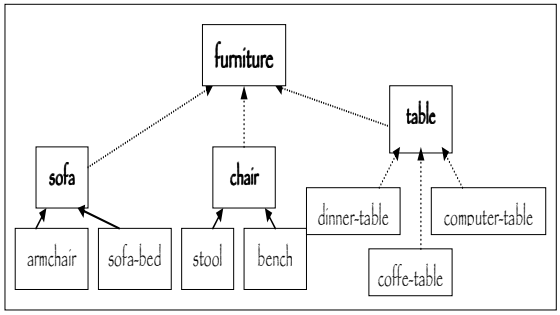

Figure 5. A fragment of the furniture taxonomy

To obtain the different types of objects we have used two existing taxonomies: one classifying different kinds of furniture, the other classifying different kinds of appliance. A fragment of the furniture taxonomy can be seen in Figure 5. For each language, the appropriate words and idioms expressing terms in those domain taxonomies have been incorporated into the list *objectype*. In the current implementation the list contains 321 entries for English, 277 for Spanish and 225 for Catalan.

4 Using context knowledge to improve efficiency

Linguistic resources adapted to the application have proved efficient when the sentences introduced by the user are limited to those supported by the grammar and lexicon. For this reason, they are specially appropriate for application-driven dialogues, where the system guides the user to introduce the data needed by the application. However, application-restricted resources can present limitations because they do not cover all possible sentences introduced by users, not even the most common mistakes.

To improve robustness the parser incorporates a garbage mechanism that eliminates all words in the user's interventions that are not covered by the linguistic resources. This garbage mechanism allows the parser to deal with spelling mistakes, ill-constructed sentences and correct sentences not covered by the domain-restricted grammars. However, it decreases the efficiency because it introduces ambiguity (different words in the user sentences can be considered "garbage") and some possibilities can lead to incorrect analyses (not corresponding to the meaning of the sentences stated by the user).

In order to limit the number of possible interpretations the parser takes advantage of the knowledge of the dialogue context. The parser obtains the text introduced by the user together with the dialogue focus. The dialogue focus consists of the attributes of the ontology concepts that have been previously asked to the user. As mentioned in Section 2, the dialogue manager uses the application specification represented in the ontology to ask the user the information needed by the application, that is, the value of the attributes describing the ontology concepts.

The approach we follow differs from that in other research dialogue systems supporting flexible communication. In dialogue systems such as (Traum et al., 1999; Meza-Ruiz and Lemon, 2005) different type of information related to the dialogue state is used to select the best hypothesis produced by the parser. This information could include current and past dialogue moves, questions under discussion, the tasks recently introduced in the dialogue, etc.

Our approach is related to that followed in many practical voice systems using VoiceXML. In the standard VoiceXML formalism the system prompt in an interaction is associated with the grammar(s) covering the next user's utterances expected. The grammar modelling the expected answer is then used by the voice engine to recognize the user's utterance. This approach has been applied successfully in practical voice systems because those systems support mainly system-driven dialogues.

In our approach the dialogue focus (the context of previous system messages) is used to give priority to the grammar rules representing the most probable user interventions in a particular state of the communication. The use of this information limits the number of rules that has to be considered at run-time and thus, it reduces ambiguity, it prevents the parser from performing semantic interpretations not useful. For this reason the run-time requirements decrease and the number of correct interpretations the parser passes to the dialogue manager (as the first correct interpretation in the list of possible interpretations) increases.

Only in the case that the user's sentence cannot successfully be analyzed using the set of rules selected, the parser would use the rest of the grammar to perform the analysis.

The selection of all grammar rules that can be involved in the expression of the information expected (the values of the attributes the system had asked the user) can be done easily because application-restricted grammars are used. In these grammars the left-hand part of the rules expressing the value of a specific attribute is represented by a category containing the prefix *gram* and the attribute identifier (i.e., in the examples shown in Figure 2 the category *gramservicetype* is the left-hand part of the grammar rule expressing the value of the attribute *servicetype* and *gramobjectype* is the left-hand part of the rules representing the *objectype* attribute).

Then, all rules that are reachable from those containing the attribute identifier in the left-hand are also selected. This process is done by applying the same top-down filtering used to select the appropriate grammar rules when processing text input. As mentioned before, the bottom-up parser uses a top-down filtering to improve efficiency. This filtering process is based on the data structure representing all reachable categories from the left-corner categories. This data structure is built previously to the analysis process.

For example, let's consider that the grammar consists of the rules in Figure 2. If the focus is *servicetype* only the first grammar rule would be selected. In the case the focus is *objectype* all rules except the first one would be selected.

5 Results and discussion

Tests have been done to compare results obtained by the parser when using the context dialogue to those obtained without this knowledge. We use the metrics precision and run-time requirements to evaluate the performance of the parser.

We have tested the parser using the grammars developed for two applications: The large Objects Collections and the Cultural Agenda, giving information about the cultural events. These grammars have been developed for three languages: English (164 rules, 437 lexical entries, 15 of them dynamic entries), Spanish (79 rules, 221 lexical entries, 15 of them dynamic) and Catalan (the translation of the rules and entries for Spanish).

In our experiments for each language we used 75 sentences introduced for a subject (native-speaker of Spanish and Catalan) not involved in the parser development. The subject answered the dialogue system prompts. The results obtained are shown in Table 1, Table 2 and Table 3. We calculated the precision of the parser results for each language. We considered the analysis of a sentence was correct when the first interpretation in the resulting list of possible interpretations was the correct one (the value of the attribute previously asked by the system). We also analyzed the decrease of the run-time requirements when using the dialogue context. We calculated the percentage of the sentences processed spending less run-time requirements (and analyzed correctly) when using context and the percentage when not using context.

	Using context	Not using context
Precision	94%	76%
Best run-time	60%	17,3%

Table 1: The parser evaluation for Spanish

	Using context	Not using context
Precision	84%	73,3%
Best run-time	45,3%	33,3%

Table 2: The parser evaluation for Catalan

	Using context	Not using context
Precision	77,3%	73,3%
Best run-time	48%	6,6%

Table 3: The parser evaluation for English

From these tables we can draw our first conclusion: using dialogue information improves precision and reduces the run-time requirements. The run-time for processing several sentences increases when context information is used. This is because selecting the grammars rules related to the dialogue focus implies some extra processing. Nevertheless, only if there is no possible ambiguity the processing without using dialogue context could be faster.

However, favouring those grammar rules expressing the answer expected by the system may limit the capabilities of the parser when processing unexpected statements. This is the case when the user's sentence gives information not only about the specific attribute the user is asked for but also about other relevant attributes. Let's consider, for example, the application for objects collection described in previous section. In order to obtain the value of the attribute *servicetype*, the system will ask "*Say what you want, information, collection or cancellation*". The user's answer could be *"Collection of two tables"*. Using context information, only the rules expressing the answer expected will be selected. In a reduced grammar those rules could only consider that the user can give information about the attribute *servicetype*. In that case, the parser would consider that the fragment "*of two tables"* is *garbage* and the resulting semantic interpretation would be *(servicetype collection)*. Then, the dialogue system would ask the user about the type of object to be collected (which could be boring for the user). In the case that all grammar rules would be considered (not only those expressing the answer expected) the whole user sentence could be correctly interpreted. For example, when using the third and fourth rules in Figure 2, the resulting semantic interpretation would be *((servicetype collection) (2, (objectype, table)))*.

In order to achieve efficient, robust and flexible communication context information must be used appropriately. Using context information to select a very limited set of rules for processing the answer expected could be useful when obtaining the value of relevant attributes (i.e., an account number) or in critical situations (when user's answers are not well understood). In other situations the context information would have to be used not only to favour the rules expressing the value of the attribute asked but also those expressing the value of other related attributes, such as those giving information about the same concept. In the example described above, both attributes *servicetype* and *objectype* give information about the concept *collection* and thus all the grammar rules related to its description would have to be selected (i.e. all the rules in

Figure 2 would be selected because all of them are related to the concept *collection*).

6 Conclusions and future work

In this paper we have described how the application specific knowledge represented in ontologies is used to improve the processing of user's interventions in a practical multilingual dialogue system for multiple applications. In this system the dialogues as well as the application-restricted resources are obtained from the ontologies representing the application specifications.

The use of ontologies has proved specially appropriate for dialogue systems in complex domains (complex concepts and relations) as well as for systems in multiple domains (information common to several applications can be easily reused) and in those systems supporting different modes of communication and languages (the semantic interpretation is represented in base to the same ontology).

We have described how a bottom-up parser uses the application specific knowledge associated with the lexical entries (obtained from the ontology representing the application) to process the user interventions. The resulting semantic interpretation is represented in base to the application specific ontology. In current implementation the semantics used to represent the application information associated with the lexical entries is very simple. For future prototypes we will study other forms of associating the application specific knowledge with the lexical entries.

In order to improve the performance the parser also uses the context dialogue. It uses information about previously asked questions (about the conceptual attribute values) to select the grammar rules related to the dialogue focus. The use of application specific and context knowledge improves both the run-time processing and the resulting semantic interpretation. However, in order to achieve a flexible communication context information must be used appropriately. Future improvements in the system will include considering different type of information related to the dialogue state.

We are planning to integrate the bottom-up parser with the voice module, using the last VoiceXML version. Future work will include comparing the results obtained when using the parser described to those obtained when using a statistical parser.

Although tests with real users have not been done yet, we expect the linguistic resources we have developed from applications specification would not cover the different ways users express themselves. For this reason, we are planning to test the current implementation of the system with different types of users and enriching the linguistic resources using the samples obtained. We are also planning to test the parser functionality in dialogues where users have more initiative.

Acknowledgements

This work has been partially supported by the EU IST FP6 project HOPS (IST-2002-507967).

We would like to thank Elisabet Comelles for her helpful comments.

References

Boullier, Pier. Guided Early Parsing. 8th International Workshop of Parsing Technologies, 2003.

Dzikovska, Myroslava. Allen, James F. and Swift, Mary. Finding the balance between generic and domain-specific knowledge: a parser customization strategy. 3rd Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2003.

Gatius, Marta. Using an ontology for guiding natural language interaction with knowledge based systems. Ph.D. thesis, Technical University of Catalonia. 2001. http://www.lsi.upc.es/~gatius/tesis.html.

Jurafsky, Daniel and Martin, James H. Speech and Language processing. Ed. Prentice-Hall, Inc. Pearson Higher Education, 2000.

Milward, David, Beveridge, M. Ontology-Based Dialogue Systems. In the Proceedings of the 3rd Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2003.

Meza-Ruiz, Ivan. and Lemon, Olivier. Using dialogue context to improve parsing performance in dialogue systems. International Workshop on Computational Semantics (IWCS), Tilburg, 2005.

Mohanty, Sanghamitra and Balabantaray, Rakesh Chandra. Intelligent parsing in Natural Language processing. 8th International Workshop of Parsing Technologies, 2003.

Traum, David, Bos, Johan, Cooper, Robin, Larson Staffan, Lewin, Ian, Mathesson, Colin, Poesio, Massimo. A model of Dialogue Moves and Information State Revision. Trindi Technical Report D2.1,1999.http://www.ling.gu.se/projekt/trindi//publications.html

Wahlster, Wolfgang . "SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell". Proceedings of the Human Computer Interaction Status Conference, 2003.