Obtaining Linguistic Resources for Dialogue Systems from Application Specifications and Domain Ontologies

Marta Gatius, Meritxell González

Talp Research Center, Software Department, Technical University of Catalonia
{gatius,mgonzalez}@lsi.upc.edu

Abstract

This paper describes the generation of application-restricted linguistic resources in different languages for a dialogue system. The dialogue system supports speech (through the telephone) and text mode (through the web) in different languages (English, Spanish, Catalan and Italian) for different applications. The text processing and voice recognition components use the same linguistic resources (grammars, lexicons and system messages). The linguistic resources are obtained from application specifications and domain ontologies.

1. Introduction

One of the challenges in developing dialogue systems for multiple applications is facilitating the generation or adaptation of the linguistic resources for a new domain. Although general resources are portable across domains they present efficiency problems. For this reason, the most common way to adapt a dialogue system to a new application is to generate the new linguistic resources necessary to express the domain concepts. The linguistic resources adapted to the communication needs of the application reduce the run-time requirements. The words in the lexicon are usually mapped to the concepts in the domain model and the parser can quickly and accurately obtain the semantic representations in the form needed by the dialogue manager. Linguistic resources adapted to the application have proven efficient especially when the sentences introduced by the user are limited to those supported by the grammar and lexicon. For this reason, they are especially appropriate for system-driven dialogues.

However, application-restricted resources are expansive to develop and difficult to reuse. Several ways have been attempted to reduce the cost of creating these resources ([1],[2],[3],[4]). Most approaches consist of isolating and representing in a declarative form the different types of knowledge involved in the communication: the domain conceptual knowledge, the knowledge controlling the interaction and the linguistic knowledge. The modular organization of the relevant knowledge into separate data structures provides great flexibility for adapting the system to different applications, languages, modes of communication and types of users.

This paper describes the generation of application-restricted linguistic resources in different languages for a dialogue system supporting two different modes of interaction: speech (through the telephone) and text mode (through the web). The dialogue system deals with multilinguality, it allows the user to choose the communication language (English, Spanish, Catalan and Italian) and it also supports interventions mixing two languages, that is, interventions in a specific language that include words (or expressions) in another language (i.e., when asking in Spanish information about an English theatre company, the company name is usually pronounced in English).

The approach we have followed when developing the system consists of obtaining the domain-restricted linguistic resources from the representation of the application knowledge involved in communication. One of the most relevant benefits of using a knowledge representation describing specific domains and services is that it facilitates a common semantic representation for different modes of interaction. In our approach, this knowledge representation is also used for obtaining the logical structure of the dialogues and the linguistic resources (grammars, lexicons and system messages) necessary for text and speech interactions in different languages.

Next section gives an overview of the dialogue system. Section 3 describes how the logical structure of dialogues can be improved by using a semantic model of the application requirements together with domain ontologies. Section 4 describes how this information can be used to obtain application-restricted linguistic resources in different languages for the speech and text modes. Finally, last section draws some conclusions.

2. An overview of the dialogue system

The dialogue system has been developed for the project HOPS ("http://www.hops-fp6.org"). The HOPS project deals with the delivery of information and online public services in the local administration. The project is focused on developing a multi-channel mass-scale e-government platform. The architecture of the platform has been designed to support an easy integration to new services. It supports interoperability between components distributed over LAN or WAN connections, mass-deployment of services for the citizens, multiple languages and operating systems, synchronous communication, interoperability behind corporate firewalls and the incorporation of new services in run-time. The architecture will support extending the number and complexity of available channels.

We have selected FADA ("http://fada.techideas.info") as the integration tool for this project because it provides integration between heterogeneous systems and applications and it does not need administration. The project is based on European standards and it is closely aligned with the work carried out by the World Wide Web Consortium (W3C, "http://www.w3.org"), that is, the VoiceXML and ontology standards. The architecture of the first prototype developed is shown in Figure 1. The components used for user interaction are described briefly next.

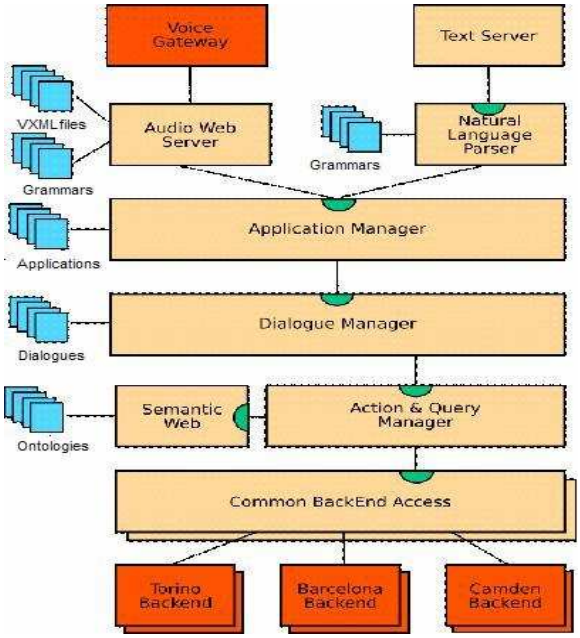

Figure 1: The architecture of the prototype

The voice components are those of the Loquendo VoiceXML platform ("http://www.loquendocafe.com"). We have selected the standard VoiceXML because it allows a rapid deployment of spoken dialogue systems to Internet-data without knowing the specific details of speech-based human-computer interaction over the telephone. That means developers in charge of adapting the system to new services do not need to be speech specialists. Besides, being VoiceXML a standard widely adopted, the applications developed using it can be adapted to several existing platforms

The user interactions are described in VoiceXML files. The VoiceXML interpreter is in charge of controlling the interactions with the user. The interpreter provides help to the user and handles particular errors that can occur in speech communication (i.e., there is no answer, the answer has not been recognized).

The automatic speech recognition system uses grammars to recognize and process user interventions. The speech recognition engine could also use statistical language modeling, but statistical language modeling has several disadvantages. The first and the most important of them is the high cost of creating a statistical language model because a significant corpus (a huge corpus for complex domains) is needed. With grammars, field data collection is useful but not indispensable. Another advantage of using grammars is that they can be easily updated when the application contents change (statistical language models are difficult to update). Besides, grammars for common recognition tasks such as date expressions, telephone numbers can be easily reused across applications.

In our system, voice grammars are represented following the Speech Recognition Grammar Specification (SRGS), in ABNF format. These grammars incorporate semantic information to facilitate the interpretation of user interventions.

The textual input is processed by a Natural Language Parser (NLP) performing syntactic-driven semantic analysis. In order to facilitate the system management, the grammars and lexicons used by the NLP are very similar to those used by the voice recognizer. The grammars used by the NLP are context-free grammars that incorporated semantic information. This semantic information in the grammar rules indicates the order of interpretation of their constituents. The lexicon entries can be words or groups of words. They incorporate semantic information obtained from the application model. The semantic formalism used is based in lambda calculus. The NLP uses XML for exchanging the information. The NLP input is an XML file containing the text introduced by the user, the language and the dialogue state. The output of the NLP is an XML file containing the resulting semantic interpretation.

The Dialogue Manager (DM) controls both speech (through the telephone) and textual interaction (through the web). Once the DM obtains the semantic interpretation from the speech and text components it plans next action. The next action can be a new interaction with the user or a query to the application back-end. The Module Action and Query Manager can be incorporated for specific services in order to optimize the back-end access. The Application Manager module has been incorporated to control the specific application/s the user requests for.

When a new interaction with the user is necessary, the DM passes the speech and text components the files containing the system messages, the language and the dialogue state. These files are created dynamically. The system messages guiding the user to introduce the information needed by the system are generated manually when adapting the system to a new application. Several of these sentences include variables which values are instantiated dynamically using information from previous sentences and/or from the application. The files generated for the speech module follows the VoiceXML specifications. These files can include the address where the grammars modeling the next expected user intervention are located. The files created for the text mode are in XML.

The prototype developed has been applied to two different types of applications: transactional and informational. The current implementation of the system supports two applications: a transactional service for collecting large objects and a service giving information about the cultural events.

3. The application knowledge

Well organized dialogues improve efficiency in communication because they help users to understand the information the service needs from them. The correct dialogue design can help minimize the number of user/system interactions and reduces the chance misunderstanding errors will occur.

In the dialogue system we have developed, the logical structure of the interaction is driven by the application specifications. The system uses a semantic model representing the data necessary to access the application and the information that can be obtained from it. This application model represents the information the system can ask/give to the user. This model is represented by a set of related concepts that are described by attributes. These attributes

correspond to the specific information about the concept that is asked (or given) to the user. The goal of the dialogues mainly consists of asking/giving the user the specific values of these conceptual attributes. The dialogue system follows the Information State Update approach [5] and the information states are obtained from the application semantic model. The sequence of interventions, the information to be provided to the users and the error recovering policies are obtained from this model.

We exemplify our approach by describing our work in the transactional service for collecting large objects from houses and companies, *The Large Objects Collection Service*. This service performs three tasks: gives information, fixes a date for collection and cancels a previous request. In the application model these three tasks are described by the concepts: *Information*, *Collection* and *Cancellation*. In Figure 2 we can see the representation of the information the system needs from the user in order to fix a date for collection. This information consists of the type of the object to be collected (represented in the attribute *object)*, whether the user is a company or a private (*userType)*, the user's name (*name)*, address (*address)* and telephone (*telephone)*. If the user is a particular, the collection is free and if it is a company, the price depends on the volume of the objects to thrown out. Once all information is obtained, the service processes the request and returns a date for collection (and a price, in case the collection is for a company). The subconcepts *Company Collection* and *Private Collection* represent the additional information that has to be exchanged for the two types of users.

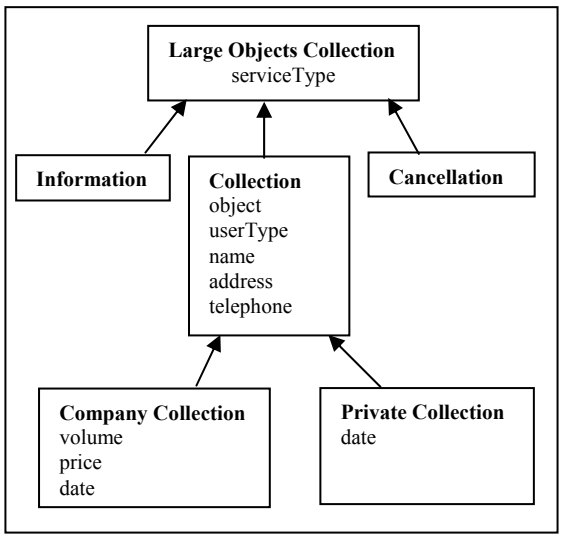

Figure 2: The Large Objects Collection service

The dialogue structure is defined using this model. Initially, the system needs information about the task required by the user. For this reason, the system will first ask the user to select one of the three tasks the system can perform (i.e., *This service gives information, fixes a data for collection and cancels a previous request. What do you want?*). Following the application model the dialogue system asks the user to give the information the application requires for performing the task. For example, if the user intervention is *I want to get ride of a table*. The system will ask the user if he/she is a particular or a company and his/her name, address and telephone.

Domain ontologies can be used when representing the application requirements to avoid asking the user difficult questions whose answer can be inferred from the ontologies. For example, in *The Large Objects Collection Service* the application needs information about whether the object to throw out is pollutant or not. This information it is not asked to the users because it could not always be correctly answered by them and it can be obtained from domain ontologies.

Domain ontologies also improve the dialogue structure by helping the system to detect an hiperonym or hipononym in the user's interventions [3]. For example, in the service for collecting large objects, the system can detect an hiperonym in the user's intervention describing the object by using the ontology classifying the different types of objects. In case the user says *I want to throw out some furniture*, because the concept *furniture* is classified in the objects ontology as an hiperonym of the information the application needs, the dialogue system would ask the user to be more specific.

4. Obtaining the linguistic resources

Using an independent base to represent the application knowledge involved in communication is specially well suited to deal with different modes (in our system, speech and text) and different languages. The same conceptual representation is used for interpreting the user intervention expressed in different modes and languages. Furthermore, the representation of the application requirements could facilitate the obtaining of the linguistic resources most appropriate for the communication. In the system we have developed, the system's messages as well as the grammars and lexicon for voice and text are generated from the representation of the application knowledge appearing in the communication.

In current implementation, the system messages asking or giving the user the conceptual attribute values are written manually. However, we are studying the possibility of using a syntactic-semantic classification of the attributes describing application concepts ([6]) to generate them semi-automatically. The syntactic-semantic taxonomy relates each attribute class to the different grammatical structures for asking and giving the attribute value. Thus, if we classify an specific attribute in the taxonomy we can obtain the grammars structures necessary to generate the sentences asking and describing the attribute values.

```
public $gramserviceType = $GARBAGE  ($GramInf1{:ret} |
   $GramC1{:ret} | $GramT1{:ret} ) $GARBAGE
                                        {<@serviceType $ret>};
private $GramInf1 =     ( information | green_point |
    reciclying_point )                  { return("information") };
private $GramC1 =       ( cancel_a_request | cancel_collection |
   cancel |   cancellation | cancelling )      { return("cancellation") };
private $GramT1 = ( fix_a_collection | collection | request| take_out |
    ask_for_a_collection_date | pick_up |)   { return("collection")};
```

Figure 3: An example of ABNF grammar for English

```
public $gramserviceType = $GARBAGE ($GramInf1{:ret} |
   $GramC1{:ret} | $GramT1{:ret} ) $GARBAGE
                                        {<@serviceType $ret>};
private $GramInf1 =    ( información | punto_verde |
    punto_de_reciclaje )                { return("information") };
private $GramC1 =      ( cancelar_una_recogida | cancelar_una_cita|
    cancelar_una_fecha | cancelar | cancelación )
                                        { return("cancellation") };
private $GramT1 =      ( fijar_una_recogida |
   solicitar_fecha_para_una_recogida |pedir_fecha | pedir_cita |
   soliciar_cita | recogida )           { return("collection") };
```

Figure 4: An example of ABNF grammar for Spanish

The grammars and lexicon necessary to process the users interventions are also obtained from the application knowledge model. These linguistic resources cover different expressions for asking and giving the values of attributes describing the application concepts. The semantic knowledge associated with these linguistic resources is defined in terms of the concepts and attributes modeling the application.

Figure 3 and Figure 4 show examples of voice grammars used for the *Large Objects Collection* service. These ABNF grammars were developed to recognize the user's answer to the first system question, asking the user to select the task to perform (attribute *serviceType* in Figure 2, which value can be *information*, *cancellation* or *collection*). Figure 3 shows the grammar modeling the user answer in English and Figure 4 shows the Spanish grammar. The semantic information incorporated into the grammar corresponds to the attribute identifier and its possible values. Notice that the semantic information associated with the English and Spanish grammars is the same.

Domain ontologies are also used to obtain the terms that represent the possible values of the conceptual attributes. For example, in the service *Large Objects Collection*, the lexical entries covering the expression of the objects to throw out have been obtained from the terms in the ontology classifying objects.

Using the application specification model favors the reusability of linguistic resources across applications. The linguistic resources (system messages, grammars and lexicons) associated with the attribute representing information required in several applications (such us a dates, addresses, etc.) are easily identified and reused.

The use of application-restricted lexicons and grammars improves accuracy and efficiency because it avoids the production of interpretations that are inconsistent with respect the application model. At each state of the communication the system guides the user to introduce the data the application needs. Thus, the speech and text processing components can use the specific grammars that most probably model the next user intervention. Furthermore, both modes incorporate a garbage mechanism that eliminates all words in the user interventions that do not express the information the system needs.

5. Conclusions and future work

In this paper we have described how application-restricted linguistic resources can be obtained from the representation of the application knowledge involved in communication. Representing the different types of knowledge involved in communication in separate bases facilitates the generation of resources for different applications, different modes of communication and different languages. We propose the use of a semantic model of the application needs and domain ontologies for improving both, the communication and the engineering process of adapting the system to new applications.

Following this proposal, we have developed a dialogue system supporting speech and text in different languages (English, Spanish, Catalan and Italian). The current implementation supports a transactional service for collecting large objects and a service giving information about cultural events. Although the system works properly when tested by developers, tests with users pointed out that it lacks of friendliness and flexibility. The linguistic resources obtained from applications do not cover the different ways users express themselves, neither the most common mistakes. For this reason, we are planning to test the current implementation of the system with different types of users (different languages, ages, skills) and enriching the linguistic resources using the samples obtained. To improve the robustness of the system we are developing new components dealing with the most frequent problems for each mode (i.e., typing mistakes in text). Future work will also be done to combine system-driven dialogues with mixed-initiative dialogues.

6. Acknowledgements

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

7. References

[1] Dzikovska, M. Allen, J. and Swift, M.. Finding the balance between generic and domain-specific knowledge: a parser customization strategy. 3rd Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2003.

[2] Hamerich, S.W.,Wang, Y. F.,Schubert, V., Schless, V. and Igel, S. XML-Based Dialogue Descriptions in the GEMINI Project. In the Proceedings of the Berliner XML-Tage 2003.

[3] Milward, D., Beveridge, M. Ontologies and the structure of dialogue. In the Proceedings of the Eight Workshop on the Semantics and Pragmatics of Dialogue Catalog, 2004.

[4] Porzel, R., Pleger, N., Merten, S., Lökelt, M., Gurevych, I. Engel, R. and Alexandersson, J. More on Less: Further Applications of Ontologies in Multi-Modal Dialogue Systems. 3rd Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2003.

[5] Traum, D., Bos, J., Cooper, R., Larson S., Lewin, I., Mathesson, C., Poesio, M. A model of Dialogue Moves and Information State Revision. Technical Report D2.1, Trindi Project, 1999.

[6] Gatius, M. and Rodríguez, H. Natural Language Guided Dialogues for Accessing the web. In the Proceeding of the 5th International Conference, Text, Speech and Dialogue, 2002.