A Platform for the Development of Spoken Dialog Systems

Emilio Sanchis, David Griol, Lluís Hurtado, Encarna Segarra
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
E-46022 València, Spain
Email: {esanchis, dgriol, lhurtado, esegarra}@dsic.upv.es

Abstract—In this paper, we present a platform for the development of dialog systems. The architecture is organized in modules, each of which accomplishes a specific objective. Communication among modules is done by means of transmission of XML data packages through sockets. This platform allows for multimodal communication and permits an easy interchange of modules, so that the change of language or task is straightforward. In our system, most of the modules use stochastic models for the representation of the different knowledge sources involved in the dialog process. These models are automatically learnt from corpora of training data. We also present a dialog system developed with this platform. The task of the dialog system presented is the access to information about train timetables and prices of trains in Spanish.

I. INTRODUCTION

One of the main objectives within the framework of human-machine speech interfaces is the development of spoken dialog systems that permit the interaction between users and computers in a natural way. Many efforts have been made in the last few years, and some prototypes for limited applications have been developed.

The main characteristics of systems of this kind are: the task is restricted to a limited domain and a medium-size vocabulary; users interact with the system in natural language (that is, spontaneous speech); the system control is made through a mixed initiative strategy (that is, the course of the dialog is not totally directed by the system, so the user can take the initiative and orient the dialog on the basis of his/her questions); the communication channel is usually the telephone line.

There are many fields of application for this kind of systems. One of the most interesting is the design of oral interfaces to access information systems, such as train timetables, flights, travel information, weather information, etc. Examples of these systems are the CMU Communicator [1], VOYAGER [2], ARISE [3], DIHANA [4], JUPITER [5]. Most of these systems are telephone-based, but there are other systems, such as speech interfaces in cars or information kiosks, which can use different channels of communication and present specific problems (i.e. the robustness against noise). Therefore, an adequate platform for the development of dialog systems must be flexible enough to be adapted to multimodal communication and changes of task, language and environment.

Due to the difficulty and the cost of developing dialog systems, it is very important to provide them with portability, allowing for changes. This is easier if the system has a high modularity and offers options for using automatic learning techniques. In a spoken dialog system, there are several modules involved. Each one of them uses its own knowledge sources and performs a specific action: a speech recognizer, an understanding module, a dialog manager, a database query manager, an answer generator, and a synthesizer. Although many works in speech recognition (and synthesis) have provided good overall results, these results are not precise enough for dialog systems. Moreover, the dialog manager design is very dependent on the task, although there are interesting proposals to facilitate the design of dialog manager strategies, such as the VoiceXML language [6].

In this paper, we propose a system architecture that facilitates the interchange of the modules of the dialog system. We have developed a platform where different modules can be allocated in different places (for example in different cities) and the communication mechanism is done by sending XML communication data packages through sockets. In this way, we can choose the most appropriate module for each situation. We have also developed some techniques for the automatic learning of stochastic models that can be used by the the speech recognizer, the understanding module, and the dialog manager [7]. We present the basic characteristics of the modules of our dialog system, including the internal aspects of their design and the data protocol defined for communication among the different modules.

This system has been developed within the framework of the EDECAN project [8], whose objective is the development of a robust, distributed, and modular dialog system for access to information systems. One of the tasks considered in this project is to provide information in natural language about train services, schedules, and fares in Spanish. This platform has been used to acquire dialogs by using the Wizard of Oh technique [9] in order to have an initial corpus of dialogs to learn the models. Once the models were obtained, we built an operative prototype.

In section 2, we describe the task; in section 3, we present an overview of the architecture of the platform; in section 4, we present the details of the different modules and interaction characteristics. Finally, in section 5, we present some conclu-
sions and future work.

II. THE TASK

The work presented in this paper is currently being developed within the framework of the EDECAN project. This project deals with the research and development of adaptation and personalization techniques in multidomain and multimodal dialog systems. One objective of the EDECAN project is the design, construction, and evaluation of a fully working prototype of a dialog system that allows the query to different information systems and the control of a media center, all within the same dialog. The system will be accessed in a home and car environment by telephone, microphone, or voice-over-IP (VOIP).

One of the tasks selected for this project is the development of an information system about train timetables and prices between the most important cities in Spain. A training corpus of dialogs of this task was acquired by means of the Wizard of Oz technique. These dialogs were acquired by telephone in Spanish. This was easily implemented in the platform because we only had to substitute the DM by a person simulating the system behavior. The acoustic models, the language models, and Dialog Manager were learnt from this corpus, and the automatic dialog system was built. The corpus consisted of 900 dialogs performed by 225 users from different Spanish regions. The size of the vocabulary was 823, and the average turns/dialog were 14. This corpus was labeled in terms of the semantic representation of the user and system utterances.

III. SYSTEM ARCHITECTURE AND COMMUNICATION DATA PACKAGES

The system architecture defined in the EDECAN project is a modular architecture that uses the client-server paradigm. Each module acts as a client of the previous module and as a server for the next one. Figure 1 shows the system architecture and the main characteristics of the important modules.

The following acronyms are used in this figure: SP (Service Provider), ASR (Automatic Speech Recognition module), SU (Speech Understanding module), DM (Dialog Manager), AM (Application Manager), AG (Answer Generator module), and TTS (Text-To-Speech synthesizer). The Service Provider is the component that is responsible for choosing the appropriate modules depending on the task, the user, and the environment.

Communication among the modules is done by sending XML communication data packages through sockets using a simplified version of the http protocol. The information about the origin module and the destination module are in the header of these data packages. All packages (except the binary ones) pass through the central communication manager, which is in charge of routing them to their destination. With this structure, the different modules do not have to know the address of the other modules, they only have to know the address of the Communication Manager. We have also defined different tags for each module to encapsulate its specific information.

Figure 2 shows an example of the XML communication data package that is used in the dialog system. This package comes from the automatic speech recognition module (ASR) and goes to the speech understanding module (SU). This information can be seen in the header of the package. Next, there is an information block that contains the sentence that has been recognized by the ASR module and the confidence measure associated to each word in the sentence. Finally, there is a tag that shows the name of the grammar used in the recognition process. We have developed two functions libraries (one in C and the other in Python) to help in the construction of new modules.

This architecture allows the connection of the Communication Manager with different modules that give the same service; for example, there may be different speech recognizers depending on the acoustic environment, or several dialog managers to deal with different sub-goals. It also might be interesting to have different answer generators in order to supply multimodal information (i.e. a voice synthesizer, avatars, or a web page generator). The switching among servers can be done in a static way (at the beginning of the dialog) or dynamically (during the dialog). At the moment, the dialog system that we have developed uses speech as input and voice synthesized as output. The selection of the different servers is made at the beginning of the dialog.
If we want to enrich the system with multimodal, multidomain functionalities and customize the system for different users, other modules will be needed, such as a user identification module, an application identification module, or an environment identification module. These modules can be easily added to the system.

IV. MODULES DESCRIPTION

A. The Communication Manager

This module is responsible for contacting the other modules in the system. It receives all of the messages sent and directs them to the destination server. It acts as a guide for the messages that are transmitted by the different modules and is responsible for establishing the communications and showing the information generated in the dialog.

In order to show this information to the user, two information blocks have been created taking into account their contents. The first block shows the control information that reflects the state of the different modules that make up the system and all the transmitted messages. The second block informs the user about the sentences that have been recognized and the answers generated by the system after each one of the interventions. Finally, there is an additional control block that shows the IP directions and ports that are used for the establishment of the sockets for each one of the modules in the system.

B. Speech recognition/understanding

The automatic speech recognition module (ASR) transforms the user utterance into the most probable sequence of words. The models that we have used in this process are Hidden Markov Models for the acoustic/phonetic modelization, and n-gram models for the language model. Although acoustic models can be trained from a general corpus of the corresponding language, in the case of language models, it is important to learn them from sentences of the task. Since errors in this phase of the dialog can create great difficulties in the following phases of the dialog, it is useful to provide the other modules with more information than only a sequence of words. The platform allows the transmission of multiple sentence hypotheses and confidence values associated to the words. This helps the DM to decide if it has to generate confirmations turns. Figure 2 shows an example of the data package between the recognizer and the understanding module.

The understanding module takes the sentence supplied by the recognition process as input and generates a semantic representation of the meaning of the sentence, which is sent to the DM. As in many other dialog systems, the semantic representation chosen for the task is based on the concept of frame (a structure with concepts and attributes).

For this task, we have defined eight concepts and ten attributes. There are two kinds of concepts: Task-dependent and Task-independent. The Task-dependent concepts represent the concepts the user can ask for, such as Hour, Price, Train-Type, Trip-Time and Services. The Task-independent concepts represent typical interactions in a dialog, such as Affirmation, Negation and Not-Understood. The attributes are: Origin, Destination, Departure-Date, Arrival-Date, Departure-Hour, Arrival-Hour, Class, Train-Type, Order-Number and Services.

An example of the semantic interpretation of an input sentence is shown below:

Input sentence:
[SPANISH] Sí, me gustaría saber el precio del tren que sale a las once.
[ENGLISH] Yes, I would like to know the price of the train that leaves at eleven.

Semantic interpretation:
(Affirmation) (Price) Departure-Hour: 11.00

Figure 3 shows an example of an XML communication data package. This package comes from the speech understanding module (SU) and goes to the dialog manager (DM).

C. The Dialog Manager

This module receives the semantic representation of the user utterance as input. It can access the database and generates an answer that is coded in terms of answer-frames. In this work, we have used two strategies to design the dialog manager module (DM). Both methodologies use confidence measures, provided by the understanding module, to determine the data reliability [10].

The first strategy is based on the use of stochastic models. One of the main characteristics of this strategy is that the system only requests explicit confirmations when the confidence measures of the words recognized is lower than a fixed threshold. This way, the interaction is more user-friendly, although the dialog can become more complicated in the case of recognition errors.

An example of a dialog performed between a user and the stochastic DM is shown in Figure 4. The data between brackets in turn U2 is a segment of words recognized with a low confidence score. Therefore, a system turn of confirmation is generated in order to be sure of the departure date.

The second DM is rule-based. It models the strategy that was defined for the acquisition of a data corpus with the Wizard of Oz technique. This strategy consists of requesting
Fig. 5. Description of the DQM.

U1: I want to know the railway timetables to go to Barcelona.
S1: When do you want to leave?
U2: I would like to travel on [May the first].
S2: Do you want to travel on 2006-05-01?
U3: Yes.
S3: There are several trains. The first train leaves at 06:23 and the last one at 21:14. Anything else?
U4: Yes, can you tell me the price of the first train?
S4: The first train is an Euromed and its price is 30 Euros. Anything else?
U5: No, thanks. Good bye.
S5: Thanks for using the information system. Good bye.

Fig. 4. A dialog using a stochastic dialog manager.

D. Answer Generator/Synthesizer

The AG translates the semantic representations of the system turns to sentences in Spanish, using templates and combining rules to make this translation. These sentences, which are in natural language, are sent to the TTS module, which makes the text conversion to voice in two phases. The first phase analyzes the input text to generate its phonetic transcription, including additional information about duration, intonation, and rhythm. The second phase processes the information that is received from the first phase and generates the suitable signals. The final waveform is obtained by means of the concatenation of voice segments previously recorded in the form of diphonemes. Finally, the waveform is modified to adapt it to the prosody of the text.

E. The Application Manager

Each different task has an AM associated to it. The function of an application manager is to control and interact with a specific application and encapsulate all task-specific details of this application. Most of the AMs are in charge of the access to a database, but there are also AMs that control devices in an intelligent environment such as a media center or a stereo equipment. When the task of the AM is to access an information system, we call it database query manager (DQM).

A relational database has been designed to store the information about train timetables and prices. The design of this database was made taking into account the specific information requirements of the EDECAN project, as well as the solutions proposed by other real systems for the same task. In the designed database, the information is structured in 11 different tables that contain information about stations, train types, ticket types, train routes, ticket prices, and user services, as well as the interrelations among all these elements.

The database was designed so that it does not contain information about individual trips. Those trips that have the same origins, destinations, schedules, and prices, are grouped together regardless of their date. This design allows us to have information about more than 400,000 independent trips in a manageable database. However, this design considerably complicates the implementation of the database queries.

The database was designed so that it does not contain information about individual trips. Those trips that have the same origins, destinations, schedules, and prices, are grouped together regardless of their date. This design allows us to have information about more than 400,000 independent trips in a manageable database. However, this design considerably complicates the implementation of the database queries.

The database query manager receives a request for information from the dialog manager as input and gives back an information structure that represents the information that has been required as a result. This module constructs the query and executes it. Once the query is executed, the DQM interprets and structures the results.

The module follows these steps:

1) The concept required for obtaining data is determined.
   The type of query to the database is determined by this concept.
2) A SQL query that is based on the requested information is constructed, using a SQL translator. If the information is not stored in a relational database, another kind of translator, such as a parsed RSS, must be used.
3) The DQM executes the query and obtains the results.
4) The information is processed to obtain the exact data.
required. This information is adapted to the requirements of the dialog manager.

5) Finally, an XML package is constructed and sent to the central communication manager.

Figure 5 shows the components and the information flow in a DQM.

V. CONCLUSIONS

In this work, we have presented an architecture to build spoken dialog systems. In this architecture, different modules perform specific functionalities in order to carry out the final goal of the system. Therefore, the correct operation and integration of each one of the components in the system is essential to ensure the correct performance of the entire system, since each module takes the result generated by the previous modules in the system. A XML protocol has been defined for the communication among the different modules.

We have described the application of this architecture in the development of a dialog system for the EDECAN project. The basic characteristics of the modules are described in detail. Different methodologies and strategies to carry out the specific functionality of the modules have been presented. The most important modules defined in the system use stochastic models that have been learnt from a data corpus.

Works which are currently being developed are the following: to improve the system architecture with the addition of new modules and functionalities; to improve the interface of the system to incorporate multimodal information; and to adapt the different modules to carry out different tasks defined for the EDECAN project.

VI. ACKNOWLEDGEMENTS

Work partially supported by the Spanish CICYT under contract TIN2005-08660-C04-02

REFERENCES


