A dialog system for the DIHANA project

D. Griol, F. Torres, L. Hurtado, S. Grau, F. García, E. Sanchis, E. Segarra

Departament de Sistemes Informàtics i Computació
Universitat Politecnica de València. E-46022 València, Spain
{dgriol,ftgoterr,lhurtado,sgraup,fgarcia,esanchis,esegarra}@dsic.upv.es

Abstract

In this paper, we present a dialog system developed in the DIHANA project. This system consists of seven modules: an automatic speech recognizer, a language understanding module, a dialog manager, a database query manager, a natural language answer generator, a text-to-speech converter and, finally, a central communication manager. For the implementation of the system, we built an architecture based on the client-server paradigm, where the central communication manager works as the client and the other modules work as servers.

1. Introduction

The study and development of automatic dialog systems is an outstanding field within the framework of language and speech technologies. A dialog system is a man-machine interface that is able to recognize and understand a spoken input and to produce an oral output as an answer. Different modules usually take part to carry out this final goal: they must recognize the pronounced words, understand their meaning, manage the dialog, perform the error handling, access the databases, and generate the oral answer.

Dialog systems that use voice as input are generally based on three fundamental aspects: (a) the communication channel is usually the telephone line, (b) the task is restricted to a certain knowledge domain and (c) 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).

In addition, natural language inputs are accepted in order to increase the system utility, by providing a wide lexicon without a rigid syntax imposed by the system (that is, allowing spontaneous speech).

The scheme used for the development of these systems usually includes several generic modules that deal with multiple knowledge sources and that must cooperate to satisfy user requirements. Some descriptions of dialog systems that are currently operative can be found in [1], [2], [3], [4], [5], [6] and [7].

The main goal of the DIHANA project [8] is the development of a robust, distributed and modular dialog system for access to information systems. In particular, we have tried to make an in-depth study of the methodological aspects in the fields of treatment of spontaneous speech, natural language modelling, language understanding, and dialog management. The task of this project is to provide information in natural language about train services, schedules, and fares in Spanish.

In this paper, we review the basic characteristics of the modules of the dialog system developed for the DIHANA project, including the internal aspects of their design and the data protocol defined for communication among the different modules.

2. System Architecture and communication data packages

As we stated above, dialog systems usually have several generic modules for dealing with different knowledge sources. To answer user queries, the coordination among the different modules is important. Figure 1 shows the system architecture.

In this figure, we use the following acronyms: ASR (Automatic Speech Recognition module), SU (Speech Understanding module), DM (Dialog Manager), DQM (Database Query Manager), AG (Answer Generator module), and TTS (Text-To-Speech synthesizer).

![Figure 1: Description of the system architecture.](image)

Communication among the modules is done by means of sending XML communication data packages through sockets. The information about the origin module and the destination module are in the header of these data packages. We have also defined specific labels for each module to encapsulate its specific information.

Figure 2 shows an 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 of the sentence. Finally, there is a tag that shows the name of the grammar used in the recognition process.

3. Automatic Speech Recognition (ASR)

The automatic speech recognition module (ASR) transforms the user utterance into the most probable sequence of words. We
used the Sphinx utilities [9] from Carnegie Mellon University to train the acoustic models and decode the user utterance.

We trained the acoustic models using the SphinxTrain program. We used semicontinuous acoustic models using 25 phonemes plus silence for the Castilian-Spanish. The acoustic training was performed using the 4 hours, 47 minutes of spontaneous speech utterances pronounced by 225 users using the Wizard of Oz technique. These utterances are part of the DIHANA corpus.

We also used a Witten-Bell discounted trigram language model constructed using the CMU-Cambridge Language modeling toolkit [10]. We used the 6,280 user utterances from the DIHANA corpus. The total number of different words was 812. We applied a categorization process with natural categories like CITY-NAME, TRAIN-TYPE, DAY, MONTH, etc. Then, the vocabulary size was reduced to 649 different words.

We used Sphinx-II as the decoder. The word error rate obtained was 14.07%.

### 4. Speech understanding (SU)

As in many other dialog systems, the semantic representation chosen for the task is based on the concept of frame. Therefore, the understanding module takes the sentence supplied by the recognition process as input and generates one or more frames, with the corresponding attributes, as output. In this task, we identified 11 concepts. Some of them are: DEPARTURE-HOUR, ARRIVAL-HOUR, FARE, etc. Each concept has a set of attributes associated to it (ORIGIN, DESTINATION, DEPARTURE-HOUR, ARRIVAL-HOUR, TRAIN-TYPE, etc). This set represents the restrictions that the user can place on each concept in an utterance.

The semantic interpretations of two input sentences are shown below:

**Input sentence:**
Quisiera los horarios de tren para ir de Valencia a Barcelona.

**Semantic interpretation:**

(DEPARTURE-HOUR)
ORIGIN: Valencia
DESTINATION: Barcelona

---

**Input sentence:**
I would like the train timetables from Valencia to Barcelona.

**Semantic interpretation:**

(DEPARTURE-HOUR)
ORIGIN: Valencia
DESTINATION: Barcelona

---

Si, me gustaría saber el precio y tipo de tren que sale a las once.

**Yes, I would like to know the price of the train that leaves at eleven and what type of train it is.**

**Semantic interpretation:**

(ACCEPTANCE)

(FARE)

DEPARTURE-HOUR: 11.00

(TRAIN-TYPE)

DEPARTURE-HOUR: 11.00

The speech understanding module (SU) is based on stochastic models estimated by means of automatic learning techniques [11]. The understanding process is done in two phases:

1. The first phase translates the input sentence into a semantic sequence defined in an intermediate language (ISL), using a stochastic model.
2. The second phase translates this semantic sequence into its corresponding frame (semantic representation, concepts and attributes, used to communicate with the dialog manager). This process is based on the use of rules that guarantee the order of the translation to frames.

The segmentation of the input sentence is made in a number of intervals that is equal to the number of semantic units in the corresponding semantic translation. Let W be the set of words in the vocabulary and V be the alphabet of semantic units. The training data consists of sets of pairs (ui,vj) that fulfill:

\[ u = u_1, u_2, \ldots, u_i, u_i = w_1, w_2, \ldots, w_{|w|}, \]

\[ w_j \in W, i = 1, \ldots, n, j = 1, \ldots, |u_i| \]

\[ v = v_1, v_2, \ldots, v_i, v_i \in V, i = 1, \ldots, n \]

Each sentence from W has an associated pair (ui,vj), where v is the sequence of semantic units and u the sequence of segments in which the original sentence has been divided. An example of the translation is shown in Figure 3.

---

**Figure 2:** Example of XML data packages.

**Figure 3:** An example of translation into the semantic language.
Once an input sentence has been segmented and a label $v_i$ has been associated to each segment $u_i$, the second phase of the understanding module consists of the translation of these pairs into one or several frames. A set of rules is applied to reorganize the contents, eliminate the useless or badly recognized information, convert quantitative values into qualitative, etc.

Figure 4 shows the result of applying the second phase of the understanding module to the example shown in Figure 3.

![Figure 4: An example of translation into concepts and attributes.](image)

**5. Dialog manager (DM)**

In this work, we present two strategies to design the dialog manager module (DM). The first strategy is based on the use of stochastic models. The states and transitions in the model are learned from the BASURDE corpus [13], which is a corpus that refers to the same task. The selection of transitions in the model is determined by the user information given in the corresponding turns, on the basis of the semantic representation generated by the understanding module.

The second manager is based on the use of rules. It models the strategy that was defined for the acquisition of a data corpus with the Wizard of Oz technique, in which a human imitates the behavior of the system. This strategy is based on the confirmation of the values provided by the user whenever the manager considers that its reliability is not high enough. Different types of data confirmation have been defined.

Both methodologies use confidence measures, which are provided by the understanding module, to determine the data reliability [14]. The use of confidence measures for error correction in the dialog management has also been proposed by other authors [15], [16].

**5.1. Stochastic dialog manager**

The behavior of the stochastic dialog system is established by the dialog manager [17]. The inputs to this module are the user semantic representations (frames) generated by the understanding module. The functions of the dialog manager are, essentially, two:

- The interpretation of their inputs, that is, the identification of the actual user dialog acts that correspond to the received user frames;
- The determination of the system strategy, mainly by the selection of the new system dialog acts (followed by the generation of their corresponding system frames).

The system frames are the output of the dialog manager module and the input to the following module, the answer generator (which translates these frames into natural language sentences).

This dialog manager determines the system strategy using two components, a stochastic dialog model (SDM) and a historic register (HR). As any dialog consists of a sequence of turns between the user and the system, the dialog manager performs the following iterative process in which it consults and updates the SDM and the HR:

1. It reads the user semantic representations (user frames);
2. It identifies the user dialog acts corresponding to the input frames;
3. It makes a transition to a user state in the SDM;
4. It updates the HR with data given by the user;
5. It makes a transition to a system state in the SDM;
6. It updates the HR in the case of querying the database;
7. It writes the semantic representation of the system turn (system frames).

The stochastic dialog model (SDM) has been learned from the BASURDE corpus (which contains 227 dialogs). It is a bigram model that includes the first and the second levels of the BASURDE dialog-act labeling. Each user (or system) state of the model is identified by a user (or system) dialog act. Thus, for each user (or system) dialog act (current state), the SDM establishes the possible transitions to system (or user) dialog acts (following states), according to the probabilities of these transitions in the corpus. Besides, we have learned another bigram model (SDM-aux), including only the first level of the corpus labeling, which is used as a back-off model in some special cases. The information of the third level of the corpus labeling is managed by means of the historic register (HR). This HR is a table that stores the values of the attributes, provided either by the user or by the database manager, and other additional information (associated confidences, confirmation state, updating time, etc.).

Currently, our dialog manager cannot follow a fully stochastic strategy. The scarceness of the training corpus causes the SDM to have a partial knowledge of the task event space. Moreover, the dialog strategy fixed by a bigram model would consider only the last user turn and would ignore all the information interchanged in the previous turns. Thus, although the system strategy always depends on the transitions in the SDM, the dialog manager needs to take into account not only the probabilities of the available transitions but also the history of the dialog (at least, the state of the values of the attributes, recorded in the HR) in order to choose the most appropriate transition. In consequence, the dialog manager follows a hybrid strategy, which is partially stochastic corpus-based, and partially fixed by a set of rules that guarantee that the system answers will be coherent with the history of the dialog.

The main mechanisms included in the dialog manager in order to complement the stochastic dialog model are:

1. The use of a process that we have called “semantic generalization” (a preprocessing of the user frames that can be considered as a kind of smoothing);
2. The use of consistence rules with the content of the HR;
3. The use of confidence measures, when these scores are provided with the input frames. For example, transitions to confirm attributes with rather high confidence scores can be weakened, while other transitions to confirm attributes with low confidence scores can be strengthened.

In [17], there is a more detailed explanation of these mechanisms.

The use of consistence rules with the content of the HR helps in looking for transitions to system states. These rules are necessary because sometimes the most appropriate system strategy is not to choose the transition with the highest probability in a bigram model (for instance, the most probable transition
could carry us to a state of asking for a certain attribute and it could be inadequate if this attribute were already known, that is, if its value was recorded in the HR).

We have implemented several rules to prune those transitions that could lead the dialog to a problematic situation (for instance, causing user misunderstanding) and these pruning rules can be seen as a common-sense interpretation of the content of the HR.

These procedures (semantic generalization, use of consistency rules, and use of the confidence measures) facilitate the adequate selection of transitions in the SDM in the majority of cases. However, some special situations where the SDM does not provide any valid transition can occur. In these situations, the dialog manager uses the back-off model (SDM-aux), the content of the HR, and the consistency rules to look for some new reasonable state in the SDM although no transition leads to it.

After including the error handling procedure in the dialog manager by means of the use of confidence measures, we evaluated the dialog system, achieving high success rates (99% and 69%), even when the received user frames had a significant error rate (20% and 30%, respectively).

An example of dialog acquired with this dialog manager is shown in Figure 5. In the figure, we use the following acronyms: U (User input), S (System output) and R (Semantic representation generated by the understanding module). Confidence scores provided by the understanding module to know the reliability (syntactic and semantic scores) are shown between brackets. In the S1 turn, an explicit confirmation of the destination and the departure date is made given their low confidence scores.

Figure 5: A dialog using the stochastic dialog manager.

5.2. Rule-based dialog manager

In our project, we also considered the development of rule-based dialog models, which are portable and adaptable to other tasks [18]. The practical implementation of these models is done by using dictionaries and files for the definition of the semantics of the task and in the determination of a standard format for the output of the dialog manager and for the communication among the modules in the system. Different models have been developed based on the strategy followed to confirm the values provided by the user whenever the manager considers that their reliability is not high enough.

The dialog manager that we present in this section models the strategy that was followed in the acquisition of the DIHANA corpus with the technique of Wizard of Oz. In this strategy, the manager interacts with the user on the basis of the levels of confidence provided by the understanding module and using a data structure that we call dialog register. The dialog register contains the current concept, attributes, and their confidence scores. The state of the dialog register and the information provided by the user in each turn are used to manage the dialog.

If all the data of the dialog register have a confidence score that is higher than the fixed threshold (safe state), the manager chooses one of the following three interactions:

- Implicit confirmation and Query to the database if the dialog register stores a frame and, at least, the values of its minimum attributes (e.g. *I am consulting the railway fares from Madrid to Bilbao in first class for you*).
- Request to the user if the dialog register does not store a value for the current concept and/or some of its minimum attributes (without a default value).
- Mixed confirmation to give naturalness to the dialog. This is made on a variable number of safe turns instead of an Implicit Confirmation-Query (e.g. *Do you want railway timetables to Valencia leaving from Granada?*).

When the state is uncertain (that is, one or more data of the dialog register have a confidence lower than the threshold), the manager selects one of the following two interactions:

- Explicit confirmation of the first uncertain item that appears in the dialog register (e.g. *Do you want to travel to Madrid?*).
- Mixed confirmation to give naturalness to the dialog. This is made on a variable number of uncertain turns of dialog instead of an Explicit Confirmation.

Finally, it should be noted that the initiative of the dialog is mixed: the information supplied by the user is taken into account even if it is not the answer to the question made by the system.

An example of dialog acquired with this dialog manager is shown in Figure 6. In the first turn of the user, no frame is detected after the user intervention. In this case, the dialog manager asks for the type of query (S1).

Once the manager confirms a frame, it verifies if there are values with a score lower than a predetermined threshold in the dialog register. A 0.5 threshold was considered for this example. The average value of the confidence scores of the DESTINATION attribute is inferior to this threshold and, therefore a confirmation of the destination value is generated (S2).

After receiving a positive confirmation, and verifying that there are no values with low reliability in the dialog register, the mandatory attributes are asked to complete the query (S3). In the following turn, after the user supplies attributes whose confidence scores are lower than the threshold, the manager selects one of the possible confirmation states (S4). If a positive confirmation is received from the user, a query to the database is made and an answer is generated facilitating the result of the query (S5).

This rule-based dialog manager was tested with 60 dialogs developed by six expert users. The understanding module provided inputs to the dialog manager with a WA of 80%.
6. Answer generator (AG)

The answer generator module (AG) translates the semantic representations of the system turns to sentences in Spanish. It uses templates and combines rules to make this translation. The input of the answer generator is composed of concepts and attributes (as in the understanding module) with confidence measures associated to each one of the frames. These measures allow the generation of detailed answers in natural language. In these answers, the attributes may or may not be mentioned depending on their associated confidence.

The technique that we use consists of having a set of templates associated to each one of the different frames, in which the names of the attributes are reflected. These names are replaced by the values obtained from the dialog register in order to generate the final answer for the user. Each frame has its set of associated templates, so that the most accurate answer is given in every possible situation for each one of the queries.

The sentence in natural language generated by the AG module is 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 diphones. Finally, the waveform is modified to adapt it to the prosody of the text.

7. Database query manager (DQM)

A database that follows the relational data model has been designed. The design of this database was made taking into account the specific information requirements of the DIHANA 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 fares, 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. This design considerably complicates the implementation of the database queries.

Most of the information included in this database corresponds with real data about trains obtained from the RENFE webpage (www.renfe.es). The implementation was made using the PostgreSQL database manager.

The database query manager (DQM) 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. Once the query is executed, it interprets and structures the results.

The module follows these steps:

1. The concept that is required to obtain 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.
3. The DQM makes the connection to the database and runs the query.
4. The information is processed to obtain the data for the specific journey. This information is adapted to the requirements of the dialog manager.

8. Central communication manager

This module is responsible for contacting the rest of the 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 distinguished 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.

9. Conclusions

A complete dialog system for information access using spontaneous speech in a restricted domain task has been presented in this work. 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 fundamental to ensure the correct performance of the complete system, since each module takes the results of this evaluation show the satisfactory operation of the dialog manager in this context, with a success rate of 93%.
result generated by the previous modules in the system. Error detection and correction techniques have also been developed. These techniques allow us to distinguish the situations in which errors appear and to make the necessary corrections to satisfactorily complete the task.

The basic characteristics of the modules that compose the DIHANA dialog system have been presented throughout the article. Different methodologies and strategies to carry out its specific functionality have been mentioned. The results obtained in the evaluation of the different components, as well as in the evaluation of the global system, are quite satisfactory.

New works are currently being developed in a new project called EDECAN [19]. The goals of these works are the following: to improve the system architecture, to make the integration of the automatic speech recognition and speech understanding tasks in only one module, to improve the graphical interface of the system and to adapt the different modules to carry out different tasks.

10. Acknowledgements

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

11. References


