A task-independent stochastic dialog manager for the EDECAN project

Francisco Torres Goterris

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Spain
ftgoterr@dsic.upv.es

Abstract

The adaptation of a stochastic dialog manager to work in a new domain is presented. A dialog manager, previously developed to attend a specific task (queries about train services), has been modified to be used in a different domain (a sport courts booking system). The new manager deals with both tasks, just loading their corresponding bigram models and configuration files. A user simulator technique has been applied to acquire a corpus, and to automatically learn the models. The dialog manager using the learnt models has been evaluated, achieving satisfactory results for using them in an acquisition with real users.

Index Terms: dialog management, stochastic models, domain independence, user simulation.

1. Introduction

The statistical approach to the design of spoken dialog systems has provided satisfactory results, as in [1], [2], and [3], and it is currently a way open for further improvements. Some drawbacks of this approach, as the high cost of the acquisition of the corpora and the evaluation made by interacting with human users, have been dealt with different strategies as, for instance, the user simulation techniques, as in [4], [5], and [6]. Equally, important efforts have been made to develop dialog systems that can be easily adapted to different domains, i.e., to obtain task-independent dialog systems, as in [7], and [8]. In this paper, the adaptation of a stochastic dialog manager to work in a new domain is explained. This adaptation is one of the objectives in the EDECAN [9] research project.

Up to now, we have developed a dialog system for the BASURDE [10] and DIHANA [11] tasks, which provides access to an information system for train timetables, prices, and services. In this system, the dialog manager [12] uses a stochastic dialog model that is a bigram model (BM) of dialog acts. The information provided in previous turns of the dialog (i.e., data out of the scope of the BM) is stored in a historic register (HR). The dialog manager selects a new state, which will determine its following action, taking into account the last user turn, the probabilities of the available transitions in the BM, and the degree of appropriateness of these transitions given the content of the HR.

In addition, we have developed a user simulator [13] that allows us to acquire synthetic dialogs, learn dialog models, and evaluate the system. The behavior of the user simulator is determined by the same BM, and by some heuristic rules that implement a collaborative dialog strategy (in order to generate consistent dialogs, which will be useful for learning dialog models). These collaborative rules are domain independent.

During an acquisition of simulated dialogs, on the one hand, the dialog manager decides its strategy using its BM and its HR, and can automatically verify the success of the dialogs and modify the BM, readjusting the probabilities of the transitions. On the other hand, the user simulator just provides an appropriate flow of user turns to easily generate consistent dialogs. This user simulation technique has been demonstrated valid to test the dialog manager and to enhance its BM [13].

In the research that is reported here, we have pointed to three aims. First, we have converted the BASURDE/DIHANA dialog manager into a task-independent dialog manager. Second, we have migrated to JAVA, developing a platform-independent prototype. Third, we have applied the user simulation technique to test the dialog manager in the EDECAN task: acquiring a dialog corpus, and learning an initial stochastic dialog model. We have obtained a dialog manager that works suitably in a dialog system for booking sport courts (the EDECAN task).

In Section 2, the EDECAN task and the design of its BM are described. In Section 3, the stochastic dialog manager is revised. In Section 4, the prototype is described, and some results of its evaluation in both tasks are reported. Finally, in Section 5, some conclusions are presented.

2. Task and dialog model description

The EDECAN project is focused on the adaptation of dialog systems to different acoustic environments and to different semantic domains. One of these tasks consists of a sport courts booking system (called the EDECAN task in this paper).

The EDECAN task has been semantically characterized by identifying the concepts and attributes involved in a set of dialogs with real users (recorded by the sport courts booking system of our University). The concepts are the goals of the user queries, and they are the following: AVAILABILITY (queries about availability of courts), BOOKING (bookings of courts, given certain restrictions), BOOKED (queries about currently booked courts) and CANCELLATION (cancellations of the bookings of courts). The attributes are the items that the user must or can provide to specify his/her goals, and they are the following: SPORT, DATE, HOUR, COURT-TYPE, and COURT-ID.

In addition, these dialogs with real users have been studied to design a set of scenarios, which are used in the acquisition of a dialog corpus. Different levels of complexity have been established in the proposed set of 15 scenarios. For instance, the first and the last of them have been coded as follows:

- Scenario-1: <AVAILABILITY> SPORT [COURT-TYPE] [DATE] [HOUR].
- Scenario-15: <BOOKED> <CANCELLATION> [SPORT] [DATE] [HOUR] [<AVAILABILITY>] <BOOKING> SPORT [COURT-TYPE] DATE HOUR.

Scenario-1 consists of a query about availability on a certain sport, allowing the user to specify date, hour, and court-type. Scenario-15 is a complex scenario, and it can be decomposed into three phases: (1) the user has to obtain the list of his/her booked courts; (2) the user has to cancel some

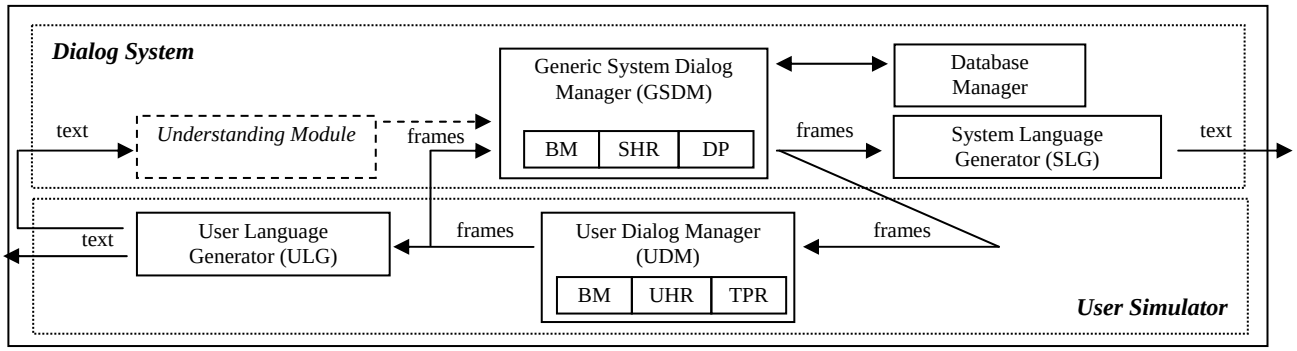

Figure 2: *Task-independent dialog system block diagram*.

courts of the previous list, and s/he can optionally provide the sport, the date, or the hour, to specify the booked court whose cancellation s/he wants; and (3) the user has to book some courts providing the sport, the date, and the hour, and s/he can supply the court-type, or can make an availability query.

Thus, dialogs of complex scenarios are dialogs composed by sequences of sub-dialogs, and there are sub-dialogs that share information among them. This circumstance occurs between the BOOKED and CANCELLATION sub-dialogs, and also between the AVAILABILITY and BOOKING sub-dialogs.

In the DIHANA project, the acquired dialog corpus was labeled applying the concept of dialog act and a hierarchy of three levels. In this hierarchy, the first level (L1) identifies the generic dialog act; the second level (L2), the semantic of the task; and the third level (L3), the instantiated attributes. Once each dialog turn was labeled, each dialog consists of a sequence of dialog acts. Thus, the structures of the dialog models are represented by sequences of dialog acts.

However, at the moment of carrying out the work reported here, there was not any EDECAN dialog corpus. Thus, after studying the corpus facilitated by our University, we have defined a set of labels for describing the semantic of the task, according to the scheme of a hierarchy of three levels. The L1 labels are the following: OPENING, CLOSING, WAITING, NEW-QUERY, QUESTION, CONFIRMATION, ANSWER, CHOICE, NOT-UNDERSTOOD, ACCEPTANCE, REJECTION. The L2 and L3 labels are the following: AVAILABILITY, BOOKING, CANCELLATION, BOOKED, SPORT, DATE, HOUR, COURT-ID, COURT-TYPE, NIL.

Using this label set, we can define the descriptors of the dialog states that will be the nodes of the BM. For instance, the (U:QUESTION:BOOKING:DATE) descriptor identifies a state in which the user asks for booking a court, specifying the date s/he wants to play. Equally, the (S:ANSWER:BOOKING:COURT-ID,HOUR) (S:CHOICE:BOOKING:NIL) descriptor identifies a state in which the system supplies a list of courts (providing their court-ids and time-slots) that can be booked, and asks the user for choosing one of them. Figure 1 illustrates this approach.

In Figure 1, the user asks for something (L1: QUESTION), the question is about bookings (L2: BOOKING), and s/he provides the values of two attributes (L3: DATE, SPORT). In the following turn, the system answers by making a confirmation (L1: CONFIRMATION) of the court-type (L2: COURT-TYPE) and it provides the value of this attribute (L3: COURT-TYPE). Then, the user carries out two dialog acts in the same turn: s/he rejects (L1: REJECTION) the court-type proposed by the system, and s/he provides (L1: ANSWER) other value of this attribute.

Starting from this labeling proposal, we have built an initial BM for the EDECAN task. The states of this BM are defined by one or several identifiers that match the (US-ID:L1-ID:L2-ID:L3-ID) pattern, where US-ID is *U* or *S* depending on the turn corresponds to the user or to the system, L1-ID is one of the L1 labels, and L2-ID and L3-ID are one or several of the L2 and L3 labels, respectively. The transitions between states have been established connecting any user (system) state to all the system (user) states. All the transitions have the same probability (i.e., given that there are *228* user states and *266* system states, the probability of the transition to any user state is *1/228*, and the probability of the transition to any system state is *1/266*). Thus, this initial BM is an equiprobable model. Given a certain current dialog state, the dialog manager will choose any following state without influence of statistical information.

Could I book a tennis-court on next Friday?
(U:QUESTION:BOOKING:DATE,SPORT)
Do you want to play on a lawn court?
(S:CONFIRMATION:COURT-TYPE:COURT-TYPE)
No. I want a clay court.
(U:REJECTION:COURT-TYPE:NIL)
(U:ANSWER:COURT-TYPE:COURT-TYPE)

Figure 1: *Labeling a segment of a hypothetic dialog*.

3. Task-independent dialog management

Spoken dialog systems are usually integrated by six modules: speech recognizer, language understanding module, dialog manager, database manager, language generator, and speech synthesizer. However, in this approach of training models through a synthetic acquisition, only the text is used, and neither speech recognizers nor speech synthesizers are part of the prototype. Figure 2 shows its block diagram.

In a synthetic acquisition, the understanding module receives the sentences generated by the user simulator, extracts its meaning, and builds a set of user frames or semantic representations. Up to now, this module is an application restricted to the BASURDE and DIHANA tasks, and it is not used in an EDECAN acquisition. Thus, the frames generated by the user dialog manager (UDM) are directly provided as input to the generic system dialog manager (GSDM), with the possibility of applying some error simulations.

The GSDM receives these user frames, decides the system dialog strategy (taking into account its BM, its system historic register, SHR, and the domain parameters, DP), and builds a set of system frames, which formalizes the chosen behavior. In addition, this manager interacts with the database manager.

The UDM receives the system frames, decides the user dialog strategy (taking into account its BM, its user historic register, UHR, and a set of target planning rules, TPR), and builds the user frames. This manager follows a collaborative strategy, defined by the TPR and a set of task scenarios.

The database manager attends the queries of the GSDM. The user/system language generators (ULG/SLG) translate the

user/system frames into Spanish or English sentences. Both modules work using a set of templates and a set of rules for instantiating the templates.

Figure 3 shows the algorithm of the dialog manager in its usual role of system interlocutor. The algorithm of the UDM (i.e., the dialog manager when it plays as user interlocutor) differs slightly from the GSDM algorithm. Both managers use the same dialog model that is a BM of dialog acts.

```
Initialization (DP, SHR);
Read (BM); BM.state = OPENING;
BM.mode = Select (STATIC, DYNAMIC);
REPEAT   Read (U-frames);
         BM.input = Adapt (SHR, U-frames);
         BM.state = Transit (BM.state, BM.input);
         SHR = Update (SHR, U-frames);
         BM.state = Transit (BM.state, SHR);
         SHR = Update (SHR, BM.state, BD.info);
         S-frames = Adapt (BM.state, SHR);
         Write (S-frames);
         IF (BM.mode = DYNAMIC) Update (BM);
UNTIL BM.state = CLOSING;
IF (BM.mode = DYNAMIC)
         Read (UHR);
         success = Compare (UHR, SHR);
         IF success Write (BM);
```

Figure 3: *System Dialog Manager (GSDM) algorithm*.

Now, we describe the steps of both algorithms, starting with the GSDM algorithm. At the beginning of each dialog, the GSDM performs the following three actions: (1) it initializes the domain parameters (DP) and the SHR, whose structure is task-dependent; (2) it reads the BM, and selects the initial state, which is the opening of dialog; and (3) it establishes the way of using the BM. In static mode, BM cannot be modified. In dynamic mode, BM can be modified when successful dialogs are carried out.

Then, and for each sequence of turns between the user and the system, the GSDM performs the following actions: (1) reading of the user frames; (2) identification of the user dialog acts corresponding to the user frames, and generation of their semantic generalizations; (3) transition to a user state in the BM, stochastically, using the semantic generalizations; (4) updating of the SHR with data from the user frames; (5) transition to a system state in the BM, taking into account the probabilities of the available transitions in the BM, and a set of heuristic rules (that check the consistence of the transitions against the content of the SHR); (6) updating of the SHR in the case of querying the database; (7) building of the semantic representation of the system turn (system frames); (8) writing of the system frames, providing them to the SLG and UDM modules; and (9), in case of working in BM dynamic mode, increasing of the counters of the chosen transitions.

The dialog ends when the closing state is reached. After this, and if it is working in BM dynamic mode, the following actions are made: (1) reading of the UHR; (2) verification of the success of the dialog by comparing both registers; and (3), in case of successful ending, the modified counters of the chosen transitions are used to recalculate the probabilities of all the transitions in the BM (which is consolidated into file).

More details about the dialog manager algorithm can be found in [12], especially in key aspects as the semantic generalization technique, and the determination of the system behavior following a hybrid dialog strategy, which is half stochastic (by using BM) and half heuristic (by using SHR).

On the other hand, the UDM performs the following actions at the beginning of each dialog: (1) it reads the DP, including the data of the scenario, and stores them in its UHR; (2) it reads the BM, and looks for a state to ask for the goals of the scenario; and (3) it generates the corresponding question frames. In each dialog turn, the UDM performs the following actions: (1) reading of the system frames; (2) semantic generalization of the frames; (3) transition to a system state in the BM, stochastically, using the semantic generalizations; (4) updating of the UHR with data from the system frames; (5) transition to a user state in the BM, heuristically, taking into account the TPR; and (6) generation of the user frames.

More details about this user simulator algorithm can be found in [13], especially the use of heuristic rules to establish the collaborative dialog strategy.

It must be remarked that both algorithms are task-independent. All the information about the tasks has been encapsulated into the bigram models, the scenarios, and other configuration files. Thus, the data-structures (models, registers) are initialized using the files that correspond to the selected task, and the methods called in both algorithms have been appropriately parameterized.

4. Development and evaluation

We have developed a JAVA platform, available in [14] as an applet, which corresponds to the design of the dialog system described in Section 3. Figure 4 shows a screenshot of this prototype in a turn of a synthetic dialog of the EDECAN task.

By means of this application, it is possible to acquire dialogs of both tasks, selecting several ways of working. In the interactive mode, any human user can provide the input frames through a graphical interface, and s/he can read the system answers, carrying out complete dialogs.

In the simulation mode, the dialog is completely done by the platform. The prototype allows us to simulate dialogs turn by turn, or whole dialogs, or series of any number of dialogs, and to specify which scenarios are simulated. In addition, the user frames can be altered by including errors in the attributes whose values are critical to achieve the success of the dialog. Moreover, there are the test and training sub-modes, which correspond to the static and dynamic modes of using the BM.

The applet area consists of seven areas of text. The three areas on the left, from top to bottom, are the real user graphical interface, the output of the UDM (user frames), and its internal state (BM transitions, and UHR content). The three areas on the right, from top to bottom, are the output of the SLG (system sentences), the output of the GSDM (system frames), and its internal state (BM transitions, and SHR content). In the bottom text area, the whole dialog is collected.

Using this prototype, we have executed several training sets for the EDECAN task, starting from the BM described in Section 2. Different trainings have been done by enabling or disabling the simulation of input errors (each training set contains 4,000 dialogs for each scenario, i.e., a total of 60,000 dialogs). Then, several test sets have been done to evaluate the learnt models (15,000 dialogs per test set). In addition, several test sets for the BASURDE task have been carried out. Table 1 summarizes the more important statistics of these test series.

Table 1. *Evaluation of the prototype in both tasks.*

Task	BASURDE		EDECAN	
Success rate	98.7	97.1	99.7	85.3
Errors per dialog	0.00	1.02	0.00	1.04
Turns per dialog	6.95	7.51	7.28	8.07

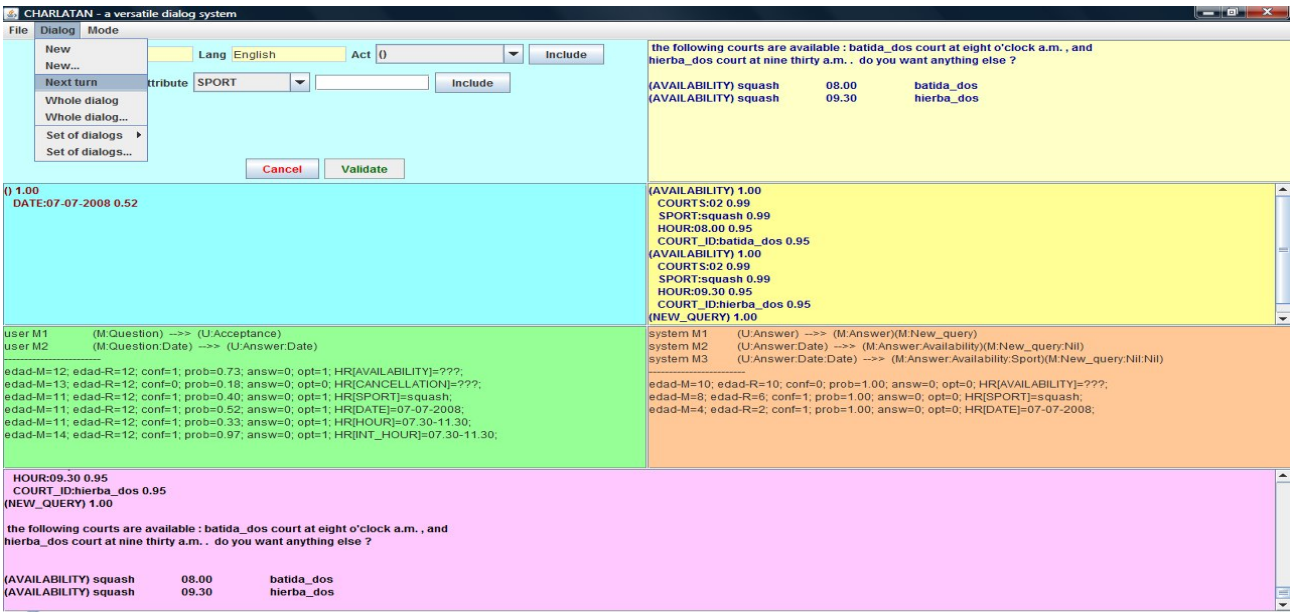

Figure 4: *The JAVA application of the dialog system.*

These results are enough satisfactory. The prototype works appropriately with the BASURDE task. In previous tests [13], a success rate of 71.8% was achieved introducing 1.12 errors per dialog (and with an average duration of 4.42 system turns). Now, the success rate has risen to 97.1% (with a lower error rate: 1.02 errors per dialog). This enhancement can be explained by the increase of the duration (7.51 system turns), which is due to a greater number of confirmations.

In addition, the prototype works finely with the EDECAN task. The success rate of 85.3% (achieved introducing 1.04 errors per dialog) is lower than the one obtained in the BASURDE test, because the EDECAN task and scenarios are more complex than the BASURDE ones. This fact also explains the higher duration (8.07 system turns).

It must be remarked that the dialog manager applies the hybrid dialog strategy [12]. However, the EDECAN training starts from an initial BM, applying a heuristic strategy. To measure the quality of the learnt model, the initial BM and the learnt BM have been tested disabling the heuristic rules. In such a situation, the initial BM does not work at all (its success rate is 5.3%), whereas using the learnt BM the success rate is 42.8% (with the same error rate). This result is coherent with a similar experiment done for BASURDE in [13].

Nowadays, the prototype can be used in both tasks. Although the success rates would be lower when interacting with real users, the working of the prototype seems acceptable to use it in a real corpus acquisition. Once this EDECAN corpus will be available, the user simulation technique can be applied to enhance the BM extracted from such a corpus.

5. Conclusions

In this paper, the adaptation of a stochastic dialog manager to deal with different tasks has been discussed. A dialog system prototype has been developed, allowing us to carry out real and simulated dialogs, acquire a synthetic corpus, learn dialog models, and evaluate the system using these models. The results are enough satisfactory as to consider using the prototype in a real acquisition with promising expectations. Thus, future work will be oriented to acquire real user dialog corpus for the DIHANA and EDECAN tasks, and to extend the prototype to another semantic domains.

6. Acknowledgements

This work has been partially funded by CICYT under project TIN2008-06856-C05-02/TIN, Spain.

7. References

[1] Levin, E., Pieraccini, R., Eckert, W., "A stochastic model of human-machine interaction for learning dialog strategies", IEEE Trans. on Speech and Audio Processing, 8 (1), 11–23, 2000.

[2] Young, S., "The statistical approach to the design of spoken dialogue systems", Cambridge University, Tech. Rep., 2002.

[3] Potamianos, A., Narayanan, S., Riccardi, G., "Adaptive categorical understanding for spoken dialogue systems", IEEE Trans. on Speech and Audio Processing, 13 (3), 321–329, 2005.

[4] Eckert, W., Levin, E., Pieraccini, R., "User modeling for spoken dialogue system evaluation", Proc. of ASRU – IEEE Workshop, Santa Barbara, USA, 1997.

[5] López-Cózar, R., De la Torre, A., Segura, J.C., Rubio, A.J., "Assessment of dialogue systems by means of a new simulation technique", Speech Communication 40 (2003), 387–407.

[6] Schatzmann, J., Georgila, K., Young, S., "Quantitative evaluation of user simulation techniques for spoken dialog systems", Proc. of SIGdial Workshop, Lisboa, Portugal, 2005.

[7] Lemon, O., Gruenstein, A., Battle, A., Peters, S., "Multi-tasking and collaborative activities in dialogue systems", Proc. of SIGdial Workshop, Philadelphia, USA, 113-124, 2002.

[8] Bohus, D., Rudnicky, A.I., "The RavenClaw dialog management framework: Architecture and systems", Computer Speech & Language (2008), doi:10.1016/j.csl.2008.10.001.

[9] Lleida, E., et al., "EDECÁN: sistema de diálogo multidominio con adaptación a contexto acústico y de aplicación", Jornadas en Tecnología del Habla (JTH), Zaragoza, Spain, 291–296, 2006.

[10] Bonafonte, A., et al. "Desarrollo de un sistema de diálogo oral en dominios restringidos", JTH, Sevilla, Spain, 2000.

[11] Benedí, J.M., et al., "Design and acquisition of a telephone spontaneous speech dialogue corpus in Spanish: DIHANA", Proc. of LREC, Genove, Italy, 1636–1639, 2006.

[12] Torres, F., Hurtado, L.F., García, F., Sanchis, E., Segarra, E., "Error handling in a stochastic dialog system through confidence measures", Speech Communication 45 (2005), 211–229.

[13] Torres, F., Sanchis, E., Segarra, E., "User simulation in a stochastic dialog system", Computer Speech & Language, 22 (2008), 230–255.

[14] Torres, F., Prototype of the dialog system, available in http://www.laesteladetanit.es/inicioB.htm.