Available online at www.sciencedirect.com
ScienceDirect
Computer Speech and Language 22 (2008) 230–255
COMPUTER SPEECH AND LANGUAGE
www.elsevier.com/locate/csl

User simulation in a stochastic dialog system

Francisco Torres *, Emilio Sanchis, Encarna Segarra

Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain

Received 28 March 2006; received in revised form 16 July 2007; accepted 4 September 2007
Available online 19 September 2007

Abstract

We present a new methodology of user simulation applied to the evaluation and refinement of stochastic dialog systems. Common weaknesses of these systems are the scarceness of the training corpus and the cost of an evaluation made by real users. We have considered the user simulation technique as an alternative way of testing and improving our dialog system. We have developed a new dialog manager that plays the role of the user. This user dialog manager incorporates several knowledge sources, combining statistical and heuristic information in order to define its dialog strategy. Once the user simulator is integrated into the dialog system, it is possible to enhance the dialog models by an automatic strategy learning. We have performed an extensive evaluation, achieving a slight but clear improvement of the dialog system.

Keywords: Dialog systems; Stochastic models; User simulation; Automatic strategy learning; Evaluation of dialog models

1. Introduction

1.1. Background literature

The research on spoken dialog systems using statistical methods has provided satisfactory results, as in Levin et al. (2000), Potamianos et al. (2005) and Young (2002). However, the high cost of an evaluation made by interacting with human users supposes a serious difficulty in the refinement of these systems. Thus, different approaches on the use of simulated dialogs as an alternative way of learning the models and evaluating the systems have been presented in recent years and can be found in Eckert et al. (1997), López-Cózar et al. (2003) and Scheffler and Young (2001). In addition, a review and discussion about the different simulation techniques and a methodology for their evaluation can be found in Schatzmann et al. (2005).

In Eckert et al. (1997) and Levin et al. (2000), the need of a user simulator is justified by the number of iterations required for automatic learning of the optimal dialog strategy. The simulation is especially appropriate in the early stages of this learning because real users would reject the initial strategy that is too deficient. The simulated user is a stochastic generative model (obtained by supervised learning from a corpus) that

* Corresponding author. Tel.: +34 96 387 73 50.
E-mail addresses: ftgoterr@dsic.upv.es (F. Torres), esanchis@dsic.upv.es (E. Sanchis), esegarra@dsic.upv.es (E. Segarra).

0885-2308/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.csl.2007.09.002

generates dialogs acts after processing the system actions. In Eckert et al. (1997), the probability distributions of the model have been computed taking into account the last system turn and the current user state (other information was not considered due to the scarceness of the training corpus). However, since this bigram model was insufficient for making an adequate prediction of the user actions, it was enhanced in Levin et al. (2000) including restrictions of the user actions in order to increase the cooperation of the user and the consistence of the dialogs. For instance, given a system request of a certain attribute, the probability of it being provided was modeled. This enhanced model is called the Levin model in Schatzmann et al. (2005).

In López-Cózar et al. (2003), a user simulator module is proposed as a tool that allows the refinement of the dialog system before real users can have access to it. The design of the user simulator requires the building of a corpus of dialog scenarios (that defines the goals of the users), and a corpus of sentences (that is adequate for dealing with the scenarios). The simulator strategy is goal-oriented and depends on the corpus of scenarios. Once the simulator decides its answer in terms of semantic representations or frames, the corpus of sentences is used to choose one sentence from the subset of sentences that correspond to the meaning of the frames. The user simulator is applied to test several confirmation strategies.

In Scheffler and Young (2001), a method for simulating mixed initiative dialogs is presented, and this technique has been applied to compare dialog strategies and to automatically learn these strategies. The user simulator has been stochastically modeled from a set of real user turns (obtained using a prototype of the system and collected in a corpus). In addition, it has been assumed that the user behavior is goal-directed. Thus, the Scheffler model combines stochastic information (in the conversational behavior modeling) and deterministic information (in the rules that fix the actions depending on the dialog goals).

1.2. Our approach

In this work, we present a new approach to user simulation to be used for the learning of stochastic dialog models. Our simulation technique has some similarities with the ones briefly mentioned above because we combine statistical and heuristic information to establish the simulated user behavior, but there are also some differences.

We have developed a dialog system for the BASURDE task (Bonafonte et al., 2000), which consists of the access to an information system for train timetables, prices, and services. In our system, the dialog manager (Torres et al., 2003, 2005a) uses a stochastic dialog model that is a bigram model of dialog acts (i.e., the states of the model represent the user and system actions). Using this model, the dialog manager selects the following state (that will determine the following system action) taking into account the last user turn and its current system state. We have not included longer contexts in the dialog model because of the scarceness of the training corpus. However, the relevant information provided in the previous turns of the dialog must be considered in order to generate consistent system actions. This problem has been solved by storing these data in a historic register and applying some heuristic rules.

The user simulator proposed in this work is a version of our dialog manager, which has been modified to play the user role. It uses the same bigram model of dialog acts. Using this model, the user simulator would select the following user action depending only on the last system action, as in Eckert et al. (1997). However, we can find some problems when the dialog manager selects the system action depending only on the last user action. Similarly, in the user simulator, there are dialog situations where the bigram model cannot provide the most appropriate user action, and the dialog history must be taken into account (through the historic register and heuristic rules). In addition, the user simulator should implement a collaborative dialog strategy (in order to generate consistent dialogs, which will later be useful for learning dialog models and will be similar to those dialogs acquired with collaborative real users). Such a strategy cannot be obtained considering only the bigram model. Thus, we have included additional information (rules and restrictions that depend on the user goals) in order to achieve the cooperation of the user and the consistence of the dialogs. These same objectives have carried to complement their stochastic dialog models with some heuristic rules in Levin et al. (2000) and Scheffler and Young (2001).

However, there are meaningful differences between the works of other researchers and our implementation of the user simulation. Our aim is to test and enhance our stochastic dialog manager; but, to achieve this, it is not necessary for the user simulator module to be properly stochastic. Thus, the user simulator module uses

the bigram model in a different way when it has to understand the last system turn and when it has to generate its new user turn. The interpretation of the last system turn is made stochastically, by comparing the semantic representation of the system turn and the dialog states to which there are transitions in the bigram model (if it is necessary, applying some smoothing that we call semantic generalization). However, the selection of the new user state (which determines the user answer and establishes the strategy) is made heuristically, according to a set of rules that define the collaborative user behavior. This asymmetry in the user simulator module does not represent any practical limitation in our objective of learning stochastic dialog models.

A corpus of acquired dialogs with real users can be used for training stochastic dialog models, even though the Wizard of Oz applied heuristic rules in the acquisition. Similarly, during an acquisition of simulated dialogs, our dialog manager uses its stochastic dialog model as a bigram model in both the interpretation of the user turn and the generation of its system turn, although the user simulator decides its answers applying heuristic rules. Our aim is to adapt or enhance the model that is used by the dialog manager. This is done through an acquisition of simulated dialogs that automatically verifies which dialogs end successfully. In these cases, we register the transitions chosen by the dialog manager, and we modify their probabilities in the model accordingly. The user simulator module is simply our tool for the easy generation of dialogs to be able to test the dialog manager and to adapt its stochastic dialog model.

Synthesizing the preceding explanations and facing our approach against the proposals of other researchers, we can underline the following important characteristic of our user simulator: the decision about its behavior (i.e., the selection of the user states) is taken according to the collaborative rules. Thus, its behavior does not depend only on the probabilities of a previously acquired corpus, and the user simulator can work using a stochastic model (learnt from a dialog corpus) or without such a model (if there is no corpus).

We have also to remark two relevant facts: first, the behavior of the user simulator depends mainly on a set of collaborative rules, and second, these rules are domain independent (because they implement generic conventions for appropriate progressing of dialogs). In consequence, our user simulator becomes a powerful tool for developing, testing and improving dialog managers. This tool can be easily adapted to be used for a different dialog task (for instance, in a dialog system for booking sport courts, as we are considering in EDECAN (Lleida et al., 2006), our current research project). Nowadays, we have not a dialog corpus of this new task of booking sport courts, and we are planning an acquisition using the Wizard of Oz technique. However, thanks to the availability of a user simulator that can work without a model trained from a dialog corpus, we can make a synthetic training of the dialog manager before the acquisition with real users. This synthetic training can provide an initial dialog model suitable for our platform of dialog acquisition.

In this paper, we describe the user simulator module and its integration in the dialog system in depth, and we also report the results of more extensive experimentation than we reported in our first article on this subject (Torres et al., 2005b). The paper is organized as follows. In Section 2, we make a brief description of the task that our dialog system deals with. In Section 3, we revise our stochastic dialog manager discussing the key aspects of its working. Next, in Section 4, we focus on the user simulator module explaining the principles of its internal design, its similarities and differences with the dialog manager, and the way of its interconnection with the other modules of the system. In Section 5, we report the results of a broad evaluation of the dialog system working in the simulation mode. Finally, in Section 6, we present some conclusions.

2. Task description

The BASURDE task consists of user telephone calls to ask for the timetables, prices, and services of Spanish trains. Four types of scenarios were defined: timetables of one-way trips, timetables of round trips, prices and services, and free scenarios. 227 dialogs were acquired using the Wizard of Oz technique. This corpus has 3528 turns (1654 user turns and 1874 system turns) and contains 14,902 words (the vocabulary size is 637). This speech dialog corpus was labeled according to a proposal (Martínez et al., 2002) based on the concept of dialog act and a hierarchy of three levels. Thus, given that each dialog turn was labeled with one or more dialog act labels, each labeled dialog consists of a sequence of dialog acts. In consequence, the structures of the dialog models learnt from this corpus are represented by sequences of dialog acts.

The first level of this labeling describes the general acts of any dialog, independently of the task (such as openings, queries, confirmations, answers, etc.). The second level is related to the semantic representation of the turn and is specific to the task (it includes concepts as departure time, price, train type, etc.). The third level represents the attributes that are instantiated in the turn (that is, there are values of these attributes in the turn). Each turn in the corpus is labeled with the identifiers of one or several dialog acts according to this proposal. Table 1 shows the labels defined for each level.

For example, a segment of a dialog can be labeled as follows:

[Spanish] *¿Me puede decir el precio de los trenes a Barcelona el sábado que viene?*
[English] Can you tell me the prices to Barcelona on next Saturday?
(U:QUESTION:PRICE:DEPARTURE-DATE,DESTINATION)
[Spanish] *¿Quiere ir de Valencia a Barcelona?*
[English] Do you want to go from Valencia to Barcelona?
(S:CONFIRMATION:DESTINATION,ORIGIN:DESTINATION,ORIGIN)

In this example, the user asks for something (first level: QUESTION), the question is about prices (second level: PRICE), and s/he provides the values of two attributes (third level: DEPARTURE-DATE,DESTINATION). In the following turn, the system answers by making a confirmation (first level: CONFIRMATION), the confirmation is about the destination and origin cities (second level: DESTINATION,ORIGIN) and it tells the user the values of these attributes (third level: DESTINATION,ORIGIN).

In the example, each dialog turn corresponds to a dialog act described with only one dialog act identifier. However, in other cases, the dialog turns are more complex and they need to be described by means of a sequence of identifiers. We can illustrate such a situation with another example:

[Spanish] *¿Quiere ir a Barcelona? ¿De dónde quiere salir?*
[English] Do you want to go to Barcelona? From where will you leave?
(S:CONFIRMATION:DESTINATION:DESTINATION) (S:QUESTION:ORIGIN:NIL)

3. Dialog manager overview

Fig. 1 shows the block diagram of the dialog system, which consists of six modules: an automatic speech recognizer, an understanding module, a dialog manager, a reply generator, a speech synthesizer, and a database manager. The behavior of our dialog system is established by the dialog manager. The inputs to this module are the semantic representations (frames) of the user turns 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 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 reply generator (which translates these frames into natural language sentences).

Table 1
Labels defined for each level

First level	OPENING, CLOSING, UNDEFINED, NOT-UNDERSTOOD, WAITING, NEW-QUERY, AFFIRMATION, REJECTION, QUESTION, CONFIRMATION, ANSWER
Second and third levels	DEPARTURE-HOUR, RETURN-DEPARTURE-HOUR, ARRIVAL-HOUR, RETURN-ARRIVAL-HOUR, PRICE, ORIGIN, DESTINATION, LENGTH-OF-TRIP, STOPS, DEPARTURE-DATE, ARRIVAL-DATE, TRAIN-TYPE, SERVICES, NIL

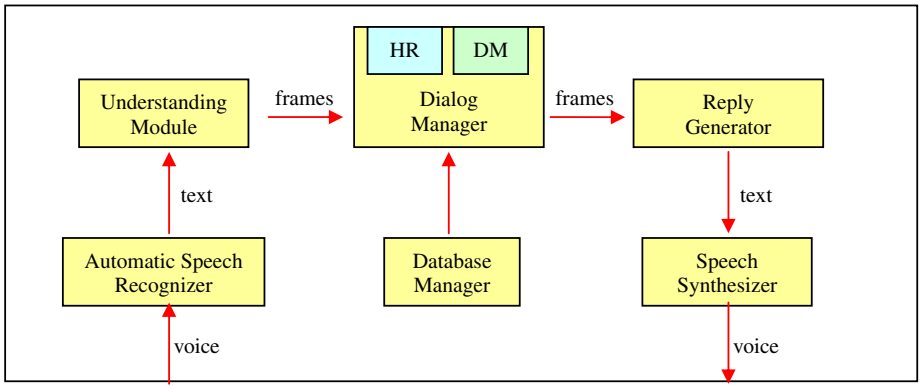

Fig. 1. Dialog system block diagram.

3.1. Algorithm of the dialog manager

Our dialog manager determines the system strategy using two components, a stochastic dialog model (DM) and a historic register (HR). The DM establishes the possible transitions between user states and system states (these states correspond to the user and system dialog acts determined in the corpus labeling). The HR stores the data interchanged between the user and the system from the beginning of the dialog. 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 DM 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 stochastic dialog model.
(4) It updates the historic register with data given by the user.
(5) It makes a transition to a system state in the stochastic dialog model.
(6) It updates the historic register in the case of querying the database.
(7) It writes the semantic representation of the system turn (system frames).

These actions can be easily identified in the dialog manager algorithm that is shown in Fig. 2.

The stochastic dialog model (DM) has been learnt from the BASURDE corpus. It is a bigram model that includes the first and the second levels of the BASURDE dialog act labeling. We have excluded the third level of the labeling because we needed to obtain a reasonable statistical model from a reduced training corpus. For instance, let us consider some possible system turns such as the following:

From where will you leave?
(S:QUESTION:ORIGIN:NIL)

From where will you leave on Monday?
(S:QUESTION:ORIGIN:DEPARTURE-DATE)

Using the DM, these turns correspond to the same dialog state identified by (S:QUESTION:ORIGIN).

Under this assumption, our DM is a more generic model. It contains fewer states, but these states are better estimated (by excluding the third level, we have reduced the number of states by 60% and the number of transitions by 50%, thereby increasing the average number of samples assigned to states and transitions). Thus, we have considered that the advantage of obtaining a more robust model in terms of probabilities estimation outweighs the disadvantage of discarding some detail in the description of the states.

In this model, each user (or system) state is identified by a user (or system) dialog act. Thus, for each user (or system) dialog act (current state), the DM establishes the possible transitions to system (or user) dialog acts (following states), according to the probabilities of these transitions in the corpus.

```
Initialization (HR);                    // storing default values in the Historic Register
Read (DM);                              // reading of the Stochastic Dialog Model
DM.state = Opening;                     // DM.state = state of DM; Opening, initial state
REPEAT
  Read (U-frames);                      // reading of the user frames                    (1)
  // identification of user dialog acts (by means of semantic generalization)          (2)
  DM.input = Adapt (HR, U-frames);
  // transition to the following user-state taking into account the received user dialog acts (3)
  DM.state = Transit (DM.state, DM.input);
  // modification of the HR with data given by frames of the user turn               (4)
  HR = Update (HR, U-frames);
  // transition to the following system-state by means of a hybrid dialog strategy,
  // taking into account the probabilities in DM and the data stored in HR          (5)
  DM.state = Transit (DM.state, HR);
  // modification of the HR with data provided by the database or from the new system state (6)
  HR = Update (HR, DM.state, BD.info);
  // generation of the system frames according to the system state and the data stored in HR (7)
  S-frames = Adapt (DM.state, HR);
  Write (S-frames);                     // writing of the system frames                  (7)
UNTIL DM.state = Closing                // Closing, final state
```

Fig. 2. Dialog manager algorithm.

In addition, we have learnt another bigram model (DM-aux), which includes only the first level of the corpus labeling. This means that (S:QUESTION) or (U:AFFIRMATION) are examples of the DM-aux states. This model 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, which can be provided either by the user or by the database manager, and other additional information (associated confidences, confirmation state, updating time, etc.).

3.2. How the dialog strategy is built

Currently, our dialog manager cannot follow a fully stochastic strategy. The scarceness of the training corpus causes the DM to have only a partial knowledge of the task event space. We could illustrate this fact by means of many dialog situations. However, let us consider only one example. In a certain dialog, the user could ask for train types, and we can suppose that (U:QUESTION:TRAIN-TYPE) is the current state in the DM. There are 13 transitions available from this state. The transitions to system states of answering the user query immediately (because there is only one type of train available for the journey) sum a probability of 0.44. The transitions to system states of answering a list of different types of trains and asking the user for choosing one of these types sum a probability of 0.32. The rest of the transitions lead to states of answering prices or timetables, or confirming the type of train, and sum very low probabilities. Thus, according to the DM, there is no transition to other system states to confirm or ask for destination, origin or departure dates. And, obviously, it seems reasonable that the dialog system could make such questions and/or confirmations before answering the user query.

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. Although the system strategy always depends on the transitions in the DM, 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. Considering again the previous example, given (U:QUESTION:TRAIN-TYPE) as the current DM state, it would be reasonable to choose the most probable transition and to answer the user query if, and only if, the values of destination, origin, and departure date are known.

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. Some important mechanisms that are included in the dialog manager to complement the stochastic dialog model are:

(a) The use of a process that we call "semantic generalization".
(b) The use of consistence rules with the content of the HR and the use of the confidence measures.

We describe these mechanisms in the following subsections.

3.2.1. Semantic generalization

Semantic generalization is a preprocessing of the user frames. The dialog manager cannot always make a direct matching between each input frame and the most similar dialog act because its corresponding user state may not be available in the DM, given the limitations of its training from a small corpus. In such a case, the semantic generalization helps in looking for transitions to user states. It can be seen as a kind of smoothing. Without semantic generalization, the dialog manager would have to compare the received user dialog act and the available user states in the DM and would transit only when an exact correspondence was found (it must be noted that the absence of transition would cause the system to lock up).

Using the semantic generalization, a set of slight formal variations of the received user dialog act is generated and all these variations are compared with the available user states in the DM, increasing the probability of finding coincidences. Moreover, when no exact correspondence is found, the coincidence criteria among the received frames and the following user states in the DM are relaxed progressively until the most appropriate state among the available user states is chosen. A more detailed explanation of the semantic generalization, including examples of the different procedures of generation of the formal variations of the received user dialog acts, can be found in Torres et al. (2005a).

We can illustrate the utility of the semantic generalization by means of an example. Let us consider the following fragment of a dialog:

[Spanish] *A Lugo, ¿en qué tipo de tren quiere viajar?*
[English] To Lugo, in which type of train do you want to travel?
(S:QUESTION:TRAIN-TYPE)
[Spanish] *No, quería ir a Vigo y en alaris, me dice horarios y precios.*
[English] No, I want to go to Vigo and with alaris, can you tell me timetables and prices?
(U:REJECTION:DESTINATION) (U:QUESTION:DEPARTURE-HOUR) (U:QUESTION:PRICE)

However, in DM, there is no transition from the (S:QUESTION:TRAIN-TYPE) state to that user state composed by a sequence of three dialog acts. Thus, without semantic generalization, it is impossible to transit in DM, and, so, the dialog manager would lock up. However, by applying semantic generalization, the dialog manager can try to transit in DM using other similar dialog acts such as (U:REJECTION:DESTINATION) (U:QUESTION:DEPARTURE-HOUR) or (U:QUESTION:DEPARTURE-HOUR,PRICE).

In this example, the chosen transition was to the (U:QUESTION:DEPARTURE-HOUR) state. And this decision is satisfactory because the system understands that the user wants timetables, it answers this query in the following turns, and later the user will have the opportunity of asking for prices again. In addition, there is no problem of discarding the meaning of (U:REJECTION:DESTINATION) in the DM transition because the provided value of destination (*Vigo*) is stored in the HR and the dialog manager will include it when it triggers the database query.

3.2.2. Selection of system states

In this subsection, we explain the search for an adequate system state in the stochastic dialog model (step 5 in the algorithm of Fig. 2) in more detail. This procedure is shown schematically in Fig. 3. As we have stated above, the use of consistence rules with the content of the HR helps in the search for transitions to system states. These rules are necessary because sometimes the most appropriate system strategy is not to choose

// transition to the following system-state by means of a hybrid dialog strategy,
// taking into account the probabilities in DM and the data stored in HR
DM.state = Transit (DM.state, HR);

Given **DM.state = s_i** *// s_i = actual user state*
HR = list {v_k, cs_k} ∀k attribute *// v_k = value of the attribute k*
// cs_k = confidence score of the attribute k
Objective:
Selection of the following DM.state = s_f *// s_f = following system state*

Procedure:
Q = Select (DM, s_i); *// list of candidates to following state*
Q = list {s_j, $p(s_i, s_j)$} where $p(s_i, s_j) > 0$
Q´ = Prune (Q, HR); *// application of the pruning rules; Q´ is a subset of Q*
Q´ = list {s_j, $p(s_i, s_j)$} where $p(s_i, s_j) > 0$ & transition ($s_i \rightarrow s_j$) is consistent with HR
Q* = Adjust (Q´, HR); *// adjustment of the weight of the allowed transitions*
Q* = list {s_j, $w(s_i, s_j)$} where $w(s_i, s_j) = \mu \cdot p(s_i, s_j) + (1 - \mu) \cdot (1 - \min_k (cs_k(s_i, s_j)))$
s_f = argmax$_j$ w(s_i, s_j) in Q*

Fig. 3. Selection procedure of the following system state.

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 this request might be inadequate if this attribute was already known, that is, if its value was already 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. The main rules are:

(1) Pruning transitions to confirm attributes if their values are already confirmed, and/or their confidence scores are above a certain threshold.
(2) Pruning transitions to answer user queries if any of the necessary attributes for building the database-query are still undetermined (e.g., without a value stored in the HR) and/or unreliable (e.g., its confidence score is below a certain threshold).

Fig. 4 shows a more formal description of these pruning rules. All the rules are domain independent. We have measured the specific contribution of these rules to the establishment of the dialog strategy. During an acquisition of 2000 dialogs in simulation mode, we have seen that the pruned transitions are the 65% of all the available transitions and the allowed transitions are the remaining 35%. Table 2 shows the percentages of application of each rule during this extensive test of the dialog system.

In consequence, when the dialog manager has to decide its reply to the user (by selecting a transition in the DM), it takes into account not only the probabilities of the transitions in the DM but also these consistence rules. The main steps of the procedure are: first, the application of the pruning rules that remove the transitions that are incompatible with the HR from the list of the possible transitions (the generation of Q', in Fig. 3); and, second, the adjustment of the weight of the allowed transitions by combining their probabilities, that are given by the DM, with the confidence scores, that are given by the HR (the generation of Q^*, in Fig. 3).

This adjustment of the probabilities is made by means of the equation:

$$w(s_i, s_j) = \mu \cdot p(s_i, s_j) + (1 - \mu) \cdot \left(1 - \min_k \left(cs_k(s_i, s_j)\right)\right)$$

where $w(s_i, s_j)$ are the weights after the adjustment, $p(s_i, s_j)$ are the initial probabilities (given by the DM), and $cs_k(s_i, s_j)$ are the confidence scores of the attributes involved in the transitions.

pruning of system answers

∀ available-transition to system-state = ***Answer*** (that implies a DB-query)
IF ∃ attribute / is-parameter-of-this-DB-query & (unknown-value OR unreliable-value)
THEN transition pruned ELSE transition allowed

pruning of system confirmations

∀ available-transition to system-state = ***Confirmation***
IF ∃ attribute / is-included-in-this-confirmation & (already-confirmed-value OR reliable-value)
THEN transition pruned ELSE transition allowed

pruning of system questions

∀ available-transition to system-state = ***Question***
IF ∃ attribute / is-included-in-this-question & (asked-value OR reliable-value)
THEN transition pruned ELSE transition allowed

pruning of reiterative actions

∀ available-transition to system-state = ***Question*** OR ***Confirmation***
IF previous system-state = ***Question*** OR ***Confirmation***
& ∃ attribute / is-included-in-both-(previous & candidate)-states
THEN transition pruned ELSE transition allowed

pruning of system "new-query"

∀ available-transition to system-state = ***New-Query***
IF ∃ attribute / is-known-value-for-a-query
THEN transition pruned ELSE transition allowed

Fig. 4. Pruning rules.

Table 2
Statistics of use of the pruning rules

	Total	%	Average per dialog
Pruned transitions to system answers	91,227	22.38	45.61
Pruned transitions to system confirmations	235,719	57.82	117.86
Pruned transitions to system questions	62,759	15.39	31.38
Pruned transitions to reiterative actions	9585	2.35	4.79
Pruned transitions to system "new-query"	8404	2.06	4.20

In this way, the probabilities of the allowed transitions to new system states are increased or decreased depending on the confidence scores of the attributes. For instance, transitions to confirm attributes with rather high confidence scores (although not high enough to be pruned) can be weakened, while other transitions to confirm attributes with low confidence scores can be strengthened. Similarly, transitions to answer user queries can be weakened or strengthened depending on the confidence scores of the attributes that trigger the query. This adjustment of the probabilities of the transitions is a temporal modification (that is, the $w(s_i, s_j)$ are not stored in DM), and, therefore, the DM remains unchanged once a transition is selected.

3.2.3. Use of the back-off model

These procedures (semantic generalization, use of consistence rules and use of the confidence measures) facilitate the adequate choice of transitions in the DM in the majority of cases. However, some special situations where the DM does not provide any valid transition can occur. In these situations, the dialog manager uses the back-off model (DM-aux), the content of the HR and the consistence rules in looking for a new reasonable state in the DM although no transition leads to it. This means that a path between two unconnected states of the DM is built by the back-off technique. However, it is a temporal path that disappears once it has been used (it does not remain in the DM).

The back-off technique can be illustrated returning to our initial example of this section: the search for a valid transition, given (U:QUESTION:TRAIN-TYPE) as the current DM state. Let us suppose a dialog situation where the value of a necessary attribute (origin, destination, etc.) is unknown. In this case, all the transitions to system states of answering the query will be pruned by applying the consistence rules with the HR. The only transition that could remain after the prune, the transition to (S:CONFIRMATION:TRAIN-TYPE), is obviously inadequate and, in the best of the cases, will lead to a situation where the user would repeat the same question.

Once all the transitions in the DM have been discarded, it is time to call DM-aux. This back-off model provides a transition from (U:QUESTION) to (S:CONFIRMATION) or to (S:QUESTION). Then, the dialog manager looks at the HR and selects some unknown or unreliable attribute. Let us suppose that it chooses the ORIGIN attribute and that DM-aux has transited to (S:QUESTION). In this case, (S:QUESTION:ORIGIN) will be the new system state in DM.

4. Dialog system with user simulation

The stochastic dialog models inferred from a scarce training corpus cover only a part of the task event space. Thus, the dialog manager using these models will have difficulties in correctly answering the real users in certain situations (those that were possible in the task but that did not occur in the corpus such as the absence of system questions or confirmations after the user state of asking for train types, that we have discussed in the previous section). We have considered the dialog simulation technique as a way of improving the stochastic dialog models. This technique allows us a quick and automatic generation of dialogs and avoids the high cost of making a new and wider acquisition interacting with real users.

Our aim is to "relearn" the stochastic dialog models by means of the simulated dialogs that end successfully. In this way, the final models will be the result of a learning that consists of two stages:

(a) A first learning from a real dialog corpus acquired using the Wizard of Oz technique.
(b) A second learning from a synthetic corpus acquired using the simulation technique.

A key question in this process is how the user behavior is modeled in the user simulator modules. Obviously, the simulator should imitate the behavior of the real users as much as possible so that the synthetic dialogs can be used to adapt the stochastic dialog models in the appropriate direction.

Fig. 5 shows the block diagram of the system extended with the user simulator modules. A user dialog manager (UDM) and a user reply generator (URG) are the components of the user simulator (the "user side" of the dialog system). These new modules are very similar to their corresponding modules on the "system side".

Both dialog managers, SDM and UDM, receive the frames from the other side, decide their following dialog actions and generate their own frames (system frames in the case of the SDM, user frames in the case of the

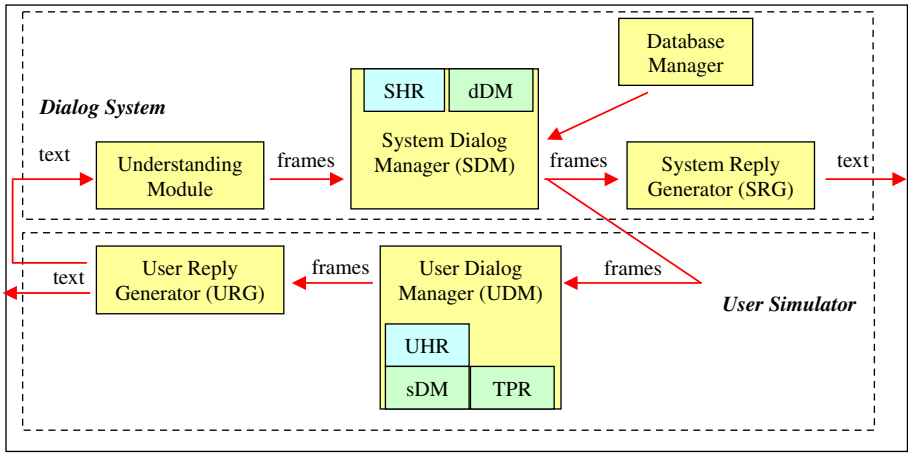

Fig. 5. Extended dialog system block diagram.

UDM). Both managers store information in their respective registers (SHR: System Historic Register; UHR: User Historic Register) and make transitions in their respective dialog models (dDM: Dialog Model dynamic version; sDM: Dialog Model static version). In addition, the UDM takes into account a set of rules (TPR, Target Planning Rules).

Likewise, both reply generators, SRG and URG, receive frames and generate natural language sentences, but the SRG processes the system frames and the URG processes the user frames (the output of the UDM). The use of the reply generators is not essential in this simulation technique because we could send the user frames from the UDM directly to the SDM (that is, without using the URG and the understanding module). However, using the URG we can obtain a simulated corpus of sentences in natural language, and not only the frames. In addition, by means of the understanding module, we can introduce errors in the frames that are sent to the SDM and, in this way, we can study the behavior of the system simulating different levels of error.

4.1. Description of the user dialog manager

Fig. 6 shows the User Dialog Manager (UDM) algorithm. This module processes the system frames, reads and writes its own historic register (UHR), reads and makes transitions in a static version of the dialog model (sDM), selects transitions according to its goals by applying a set of rules (TPR), and generates the user frames.

At the beginning of each dialog, the UDM performs the following actions:

(1) It reads the parameters and objectives of the scenario and stores this information in the UHR.
(2) It reads the model (sDM) and looks for a state to ask about one or more objectives.
(3) It generates the question frames.

In each dialog turn, the UDM performs the following actions:

(4) It reads the system frames.
(5) It compares these frames with the possible system dialog acts (semantic generalization).
(6) It transits to a new system state in the sDM.
(7) It updates the UHR with data given by the system.
(8) It transits to a new user state in the sDM.
(9) It generates the user frames.

At the end of the dialog, the UDM writes the UHR (step 10 in Fig. 6) because the SDM needs it to verify the success of the dialog.

Some details of the UHR initialization must be mentioned. At the beginning of the dialog, the attributes (which are parameters and objectives of the scenario) are read and stored in the UHR. These attributes always

```
Initialization (UHR);                          // UHR = User Historic Register      (1)
Read (sDM);                                    // sDM = static Dialog Model         (2)
sDM.state = Question;                          // sDM.state = state of sDM          (2)
U-frames = Adapt (sDM.state, UHR);             // user frames of initial state      (3)
Write (U-frames);                                                                   (3)
REPEAT
  Read (S-frames);                             // reading system frames             (4)
  sDM.input = Adapt (UHR, S-frames);           // semantic generalization           (5)
  sDM.state = Transit (sDM.state, sDM.input);  // transition to system-state         (6)
  UHR = Update (UHR, S-frames);                // HR update by system turn          (7)
  sDM.state = Transit (sDM.state, UHR, Rules); // transition to user-state           (8)
  U-frames = Adapt (sDM.state, UHR);           // building user frames              (9)
  Write (U-frames);                            // writing user frames               (9)
UNTIL sDM.state = Closing;
Write (UHR)                                                                         (10)
```

Fig. 6. UDM algorithm.

have the same values for each one of the scenarios. This could cause that different simulations of a given scenario were excessively similar, not allowing an adequate exploration of the task event space.

We have solved this problem by means of confidence scores. The UDM can include or exclude the values of the attributes in their turns depending on the confidence scores of the attributes. We can extend the range of possible questions and answers of the UDM through a random generation of the confidence scores. Thus, we start any simulation by reading the values of the attributes of the scenario and giving random values to the confidence scores of these attributes. In this way, the UDM is able to include different subsets of the attributes in their turns of different simulations of the same scenario. This procedure allows us to reasonably explore the task event space with a reduced set of scenarios.

Another aspect of the UDM operation that needs to be explained is how it implements a collaborative dialog strategy. The main difference between the SDM and the UDM algorithms is in the procedure of choosing transitions in its own turn:

(a) When the UDM processes the received frames (in the system turn), it uses the model as a bigram model and applies semantic generalization just as the SDM does.
(b) However, in the user turn, the UDM uses the model as a whole list of possible new user states and does not take into account whether there are transitions from the current system state to them. Instead, it applies a set of rules to look for the best new user state, according to the current content of the UHR and a collaborative strategy. This strategy has been defined to satisfy the aims of the simulated scenario.

Fig. 7 shows the main rules of this collaborative strategy for choosing a user state. The first rule establishes that, after a system answer, the UDM verifies the existence of objectives that are still unknown and, depending on this, it transits to either a question state or the closing state.

The second rule establishes that, after a system question about unknown attributes, the UDM searches for the values of the requested attributes and, if it finds them (because the simulated scenario gives them values), it transits to an answer state. Otherwise, the transition is made to a question state of an objective attribute. In this last case, there is still some probability of satisfying part of the system question by providing some data spontaneously.

management of system answers (first rule)

IF system-state = *Answer*
THEN look for new targets in UHR
  IF ∃ attribute / is-target & unknown-value
  THEN (look-for & transit-to) user-state = *Question* of the attribute
  ELSE (look-for & transit-to) user-state = *Closing*

management of system questions (second rule)

IF system-state = *Question*
THEN look for asked attribute in UHR
  IF attribute / known-value
  THEN (look-for & transit-to) user-state = *Answer* of the attribute
  ELSE (look-for & transit-to) user-state = *Question* of a target attribute

management of system confirmations (third rule)

IF system-state = *Confirmation*
THEN look for asked attribute in UHR
  IF attribute / known-value
  THEN IF attribute-value(UHR) = attribute-value(frames)
    THEN (look-for & transit-to) user-state = *Affirmation* of the attribute
    ELSE (look-for & transit-to) user-state = *Rejection* of the attribute
  ELSE (look-for & transit-to) user-state = *Question* of a target attribute

Fig. 7. Rules applied by the UDM.

The third rule establishes that, after a request for attribute values confirmation, the UDM verifies that the values of the requested attributes are known, compares them with those hypothesized by the system and, depending on their equality, it transits to either an affirmation state or a rejection state. Again, if the requested attributes are not known, it transits to a question state of an objective attribute.

4.2. Modifications in the system dialog manager

Fig. 8 shows the new System Dialog Manager (SDM) algorithm, i.e., a modified version of the one previously shown in Fig. 2. The main change is that the new dialog manager can modify its stochastic dialog model. When the new dialog manager is running a synthetic acquisition, it can change the probabilities of the transitions in the model depending on the success of the simulated dialogs. After a successful simulation, the probabilities of the selected transitions are increased (reducing the probabilities of other possible but unselected transitions). In addition, by joining unconnected states of the model, the new dialog manager can create new transitions, when it uses the back-off technique (making extraordinary transitions as explained in Section 3.2.3) and the simulation ends satisfactorily. Thus, by using this SDM, the stochastic dialog model can be adapted dynamically, that is, at the same time that the acquisition is made.

As can be observed in Fig. 8, the principal changes are:

(a) In each dialog turn, the selected transitions are temporarily added to the model, by readjusting the probabilities accordingly (steps 1 and 2 in Fig. 8).
(b) At the end of the dialog, the UHR is read, and the two registers (UHR, SHR) are compared to determine the success of the simulation (step 3 in Fig. 8). In the case of a successful ending, all the transitions that were temporarily added to the model are permanently added (step 4 in Fig. 8). Otherwise, they are removed, leaving the model as it was at the beginning of the dialog.

The updating of the probabilities is easily made because the model's file stores the frequency counters of all the transitions. Thus, after a successful simulation, the frequency counters of the selected transitions are increased, stored in the file, which is read again (at this moment, all the probabilities are recalculated).

Fig. 9 shows the algorithm used for the automatic evaluation of the success of the simulations. The possible reasons for dialog failure are the following:

```
Initialization (SHR);                          // SHR = System Historic Register
Read (dDM);                                    // dDM = dynamic Dialog Model
dDM.state = Opening;                           // dDM.state = state of dDM
REPEAT
  Read (U-frames);                             // reading user frames
  dDM.input = Adapt (SHR, U-frames);           // semantic generalization
  dDM.state = Transit (dDM.state, dDM.input);  // transition to user-state
  dDM = Update (dDM, transition);              // temporal modification of dDM      (1)
  SHR = Update (SHR, U-frames);                // HR update by user turn
  dDM.state = Transit (dDM.state, SHR);        // transition to system-state
  dDM = Update (dDM, transition);              // temporal modification of dDM      (2)
  SHR = Update (SHR, dDM.state, BD.info);      // HR update by system turn
  S-frames = Adapt (dDM.state, SHR);           // building system frames
  Write (S-frames);                            // writing system frames
UNTIL dDM.state = Closing;
Read (UHR);
success = Compare (UHR, SHR);                                                    (3)
IF success THEN Write (dDM, transitions)       // permanent modification of dDM   (4)
```

Fig. 8. SDM algorithm.

```
IF (number-of-turns>T-max) THEN Failure ("duration")
IF ∃ attribute / attribute-value(UHR) ≠ attribute-value(SHR)
   IF attribute / is-parameter THEN Failure ("parameters")
   IF attribute / is-target    THEN Failure ("targets")
IF ∀ attribute / attribute ∈ scenario
          & attribute-value(UHR) = attribute-value(SHR)
THEN Success; Write (dDM, transitions)
```

Fig. 9. Evaluation algorithm of simulation success.

(a) The simulation is considered unsuccessful if it has an excessive duration (a number of turns that is higher than a given threshold) because real users do not maintain very long dialogs (the threshold establishes the limit of the user's patience).
(b) The simulation ends unsuccessfully if there is any difference in the values of the attributes that define the user query because, in such a case, the user has received incorrect information (the system has answered a query that is not the query made by the user).
(c) And, finally, the simulation does not conclude satisfactorily if there is any difference in the values of the attributes that are the answer of the system (that is, the system has correctly understood the user query and answered it, but some errors can occur in the user simulator understanding phase and cause incorrect values to be recorded in the UHR).

In consequence, the equality between all the slots in both historic registers implies the success of the simulation.

4.3. Possibilities of the simulation technique

It must be noted that the simulation technique not only increases or decreases the probabilities of the transitions but it can also add new transitions to the stochastic dialog model. In order to explain this capability, we have to emphasize the difference between ordinary model transitions and extraordinary model transitions, given how the dialog manager uses its model.

A model transition is called *ordinary* when the dialog manager transits from a current system (or user) state to a new user (or system) state following a non-zero probability arc between these two states (that is, following a transition that actually exists in the stochastic model). In this case, the dialog manager is able to match the situation of the dialog that is running with a dialog act bigram (*current-state* → *new-state*) that exists in the training corpus. Therefore, it can use the DM as a bigram model.

A model transition is called *extraordinary* when the dialog manager transits from a current system (or user) state to a new user (or system) state following a zero probability arc between these two states (that is, the stochastic model does not contain such a transition). This occurs when the dialog manager cannot find the necessary dialog act bigram (*current-state* → *new-state*) because it does not appear in the training corpus and, so, it does not exist in the model. As we have already mentioned in Section 3, the dialog manager applies a back-off mechanism (that combines the auxiliary model, DM-aux, and the history of the dialog, SHR) that provides a way of managing those dialog situations that are unseen in the corpus and, so, are absent in the main dialog model, DM.

In the normal operation of the dialog system (when interacting with real users as in Fig. 1), the back-off procedure avoids the lock-up of the dialogs when it is called by allowing "jumps" between unconnected states of the DM. However, we do not know whether these jumps (transitions that do not exist in the model but that are temporarily created as an emergency solution) would be, or not, good solutions, in the sense of carrying the dialogs to satisfactory endings. Since it is not possible to compare the content of the SHR and the beliefs of the users at the end of each dialog, the validity of the dialogs cannot be determined.

On the other hand, using the dialog system in simulation mode (Fig. 5), we can automatically make the comparison between the contents of the two registers (UHR and SHR). Thus, we can decide which new

extraordinary transitions (generated using the back-off mechanism) are useful and which are not. In addition, by means of the generation of thousands of dialogs, the simulation mode allows us to explore the event space quickly and without cost.

In summary, an extraordinary transition can be converted into an ordinary transition during the training of the model by means of the simulation technique. This change occurs each time that a simulation, in which extraordinary transitions have been made, ends successfully. In consequence, the simulation technique allows us to extend the knowledge of the task event space by adding new transitions that correspond to dialog situations that did not occur in the BASURDE corpus.

5. Evaluation

We defined a set of scenarios, which consists of queries about timetables, prices and/or train types for one-way trips. Table 3 shows the specifications of the 14 scenarios that we have considered. In Table 3, each row specifies one scenario, and each column specifies the values of an attribute. The "???" string is used to indicate that the attribute is a target of the scenario. The "---" string is used to indicate that the attribute is not relevant in the scenario (that is, it can take any value). Any other string indicates the value of the attribute in the scenario (that is, it establishes a restriction or parameter of the user query).

We ran several series of a few thousand simulated dialogs using this set of scenarios and varying some parameters in the experimentation (including or not including ASR and/or understanding errors in the simulations). In this way, we obtained new stochastic dialog models (always starting from the DM inferred from the BASURDE corpus, but with each one being adapted to different experimental conditions). Then, we tested these new models under the same conditions used for training (same rate of ASR and/or understanding errors) and we compared their performance with the initial model, the BASURDE model.

We used the user simulator modules for both the training of the new models (called DM^*, DM^{*e}, DM^{*f} and DM^{*q}) and for the comparative test between the initial model and the new models. During each test, the system and the user simulator were working with the same dialog model. For instance, in the test of the DM^* performance, both dialog managers (the SDM and the UDM) used the same model, DM^*.

5.1. Training of the models through simulation

First, we adapted the initial model, DM, running 5000 simulations without errors in their inputs (that is, considering that the user input was perfectly recognized). Let DM^* be the model obtained after this training. Table 4 shows some statistics from the DM^* training. Given that we did not include errors and that the user simulator followed a collaborative strategy, almost all the dialogs ended successfully (only 1.38% of the

Table 3
Set of scenarios

	Origin	Destination	Train type	Price	Departure date	Departure hour	Arrival hour	Departure hour interval	Arrival hour interval
1	Vigo	Soria	Alaris	???	24/09/---	???	---	13.00–21.00	---
2	Toledo	Salamanca	---	---	04/11/---	???	???	05.00–13.00	---
3	Bilbao	Sevilla	---	???	24/12/---	???	---	13.00–21.00	---
4	Zamora	Granada	Estrella	???	22/03/---	???	???	09.00–12.00	17.00–21.00
5	Valencia	Zaragoza	???	---	30/06/---	???	???	05.00–10.00	10.00–13.00
6	Albacete	Madrid	---	???	08/07/---	???	???	04.00–09.00	07.00–10.00
7	Valencia	Lérida	---	---	28/09/---	???	???	---	---
8	Lugo	Valencia	---	???	20/03/---	???	???	---	---
9	Barcelona	Bilbao	Talgo	---	22/11/---	???	???	---	---
10	Valencia	Sevilla	---	???	31/07/---	---	---	---	---
11	Valencia	Alicante	---	---	06/12/---	???	???	---	---
12	Valencia	Madrid	Talgo	???	11/11/---	???	???	04.00–10.00	07.00–12.00
13	Alicante	Salou	---	---	21/09/---	???	???	10.00–13.00	16.00–20.00
14	Monzón	Lérida	---	???	15/10/---	???	???	07.00–10.00	10.00–13.00

Table 4
Some statistics of the DM* training

DM* training	Total	%	Average per dialog
Dialogs	5000		
Successful dialogs	4931	98.62	
System turns	23,626		4.73
Answer system turns	11,897		2.38
Question and/or confirmation system turns	10,958		2.19
Other system turns	558		0.11
Ordinary transitions to user states	23,574	99.78	
Extraordinary transitions to user states	52	0.22	
Ordinary transitions to system states	23,413	99.10	
Extraordinary transitions to system states	213	0.90	

dialogs failed because of their excessive duration). In these experimental conditions, excessive duration is the only possible reason of failure, according to our method of measuring simulation success (Fig. 9).

In Table 4, as in the following tables of this section, we have classified the system turns in three categories: answer states, question and/or confirmation states, and other states. The first two categories contain the dialog acts that are useful for the dialog progress (states where the system provides information, asks for information, or wants to confirm information). The last category (other states) gathers a miscellaneous set of states (mainly, states of wait and/or not understanding) that do not help in the progress of the dialogs. We have to indicate that each dialog always begins with a system opening state and ends with a system closing state, but we have preferred not to include these opening and closing states in the statistics (if the transitions to these states were considered, we had just increased the average number of turns by 2).

In Table 4, we have also distinguished between ordinary and extraordinary transitions in the model. It can be observed that the extraordinary transitions have low percentages of occurrence.

Fig. 10 shows the evolution of the probabilities of some transitions during the training. We have chosen these transitions because they illustrate some interesting changes:

- There are reasonable transitions with significant probabilities in DM*, whereas they do not exist (i.e., they have zero probability) in the initial model, DM. In these cases, extraordinary transitions are converted into ordinary transitions during the training. Such a conversion occurs, for instance, when a transition is made

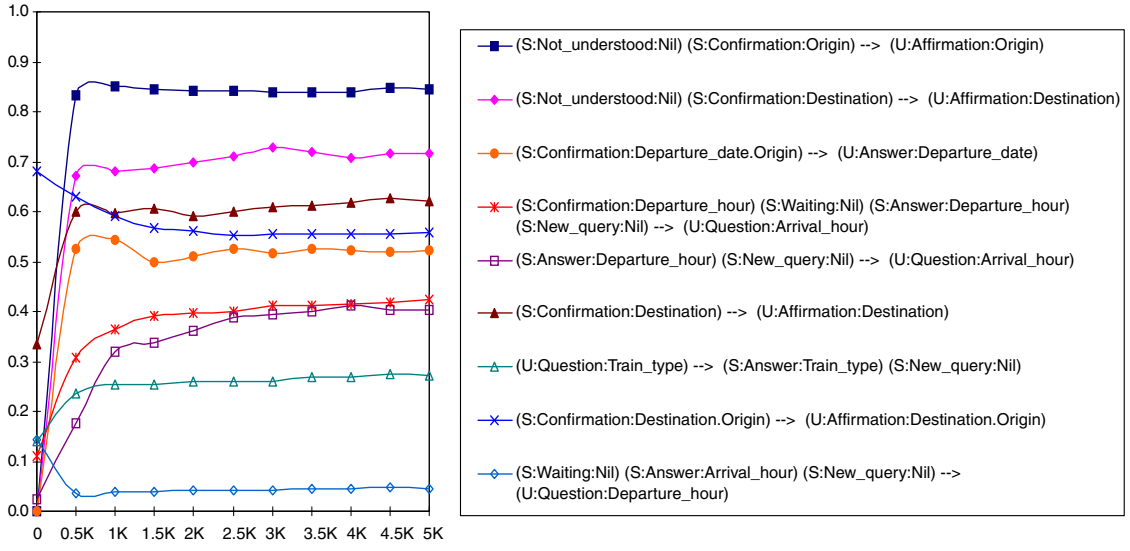

Fig. 10. Evolution of the probabilities of some transitions during the DM* training.

from a system state that confirms an origin to a user state that accepts this origin (from the (S:NOT-UNDERSTOOD:NIL) (S:CONFIRMATION:ORIGIN) state to the (U:AFFIRMATION:ORIGIN) state, in Fig. 10). Another case is when a transition is made from a system state that confirms a departure date and an origin to a user state that answers the departure date (from the (S:CONFIRMATION:DEPARTURE-DATE,ORIGIN) state to the (U:ANSWER:DEPARTURE-DATE) state, in Fig. 10).

- There are other reasonable transitions that appear in both models, but with a higher probability in the DM*. An example is the transition from a system state that answers departure timetables to a user state that asks for arrival timetables (in the figure, from the (S:ANSWER:DEPARTURE-HOUR) (S:NEW-QUERY:NIL) state to the (U:QUESTION:ARRIVAL-HOUR) state). This modification can be easily explained, since many scenarios have departure and arrival timetables as targets and the user simulator is programmed to ask for both targets in a certain order (asking for the arrival time once the departure time has been obtained, applying a common sense rule).
- There are transitions with a lower probability in the DM* than in the DM. Obviously, the transitions that are not selected during the simulations, or that are selected very few times, decrease their probabilities. This occurs when a transition is made from a system state that answers arrival timetables to a user state that asks for departure timetables (in the figure, from the (S:WAITING:NIL) (S:ANSWER:ARRIVAL-HOUR) (S:NEW-QUERY:NIL) state to the (U:QUESTION:DEPARTURE-HOUR) state). Generally, when the system provides an arrival time, the user simulator has already obtained the departure time, and it does not need to request it.

In Fig. 10, it can be observed that the probabilities tend to converge after running over 1000 simulations. Therefore, the simulation technique allows an adaptation of the DM that is dynamic and stable, because it converges after a moderate number of iterations.

In the second training, we adapted the initial model, DM, running 5000 simulations with errors in their inputs (i.e., introducing recognition and/or understanding errors). Let DM*e be the model obtained after this training. Table 5 shows some statistics of the DM*e training. Since we included errors, many dialogs ended incorrectly (28.14% of the dialogs).

We also observed the convergent evolution of the probabilities of the transitions during this new training. Given the existence of erroneous values of the attributes, the probabilities of the transitions from a system state confirming one or more attributes to a user state accepting them converge to lower values in the DM*e model than in the DM* model.

To compare the available transitions from a given state in the two models is another way of showing the modification of the dialog strategy through the adaptation of the stochastic dialog model. We can consider two examples: the transitions from two important user states, (U:ANSWER:DEPARTURE-DATE) and (U:QUESTION:DEPARTURE-HOUR).

Table 5
Some statistics of the DM*e training

DM*e training	Total	%	Average per dialog
Dialogs	5000		
Successful dialogs	3593	71.86	
Dialogs without errors	2295	45.90	
Dialogs with errors	2705	54.10	
Errors in ORIGIN attribute	1878		0.37
Errors in DESTINATION attribute	2080		0.42
Errors in DEPARTURE-DATE attribute	1626		0.33
System turns	22,101		4.42
Answer system turns	11,997		2.40
Question and/or confirmation system turns	9834		1.97
Other system turns	270		0.05
Ordinary transitions to user states	22,089	99.95	
Extraordinary transitions to user states	12	0.05	
Ordinary transitions to system states	21,988	99.50	
Extraordinary transitions to system states	113	0.50	

There are nine transitions from the (U:ANSWER:DEPARTURE-DATE) state in DM, the initial BASURDE model. The most probable transition (with a probability, $p = 0.53$) leads to a system state that answers departure timetables. The second most probable transition ($p = 0.25$) leads to a system state that answers departure timetables for the return journey of a round trip. Other transitions lead to different confirmation states with lower probabilities (values around 0.02).

In the new model, DM^{*e}, there are 38 transitions from the (U:ANSWER:DEPARTURE-DATE) state and 18 of these transitions have probabilities that are higher than 0.01. The comparison of the lists of transitions in the DM and DM^{*e} models shows clear changes in the available options for the system behavior (i.e., a modification of the dialog strategy depending on the dialog model used).

In DM^{*e}, the most probable transition also leads to the same system state that answers departure timetables, but its probability decreases significantly (from 0.53 to 0.40). The change is even more abrupt in the case of the transition to answer departure timetables for the return journey (from 0.25 to 0.01). This is because we have not considered scenarios for round trips in our simulations. On the other hand, transitions to confirmation states are more probable in DM^{*e}: the probability of confirming a departure date increases from 0.02 to 0.03, the probability of confirming an origin increases from 0.02 to 0.17, the probability of confirming a destination increases from 0.02 to 0.05, etc. In addition, DM^{*e} contains transitions to states that answer prices or arrival timetables (with very low probability), while such transitions do not exist in the initial model.

We can make another similar comparison taking into account the (U:QUESTION:DEPARTURE-HOUR) state. In the DM model there are 22 transitions from this state whereas the new model provides 42 transitions from it. The most probable transitions in DM lead to states that answer departure timetables (probability = 0.47), states that confirm an origin and a destination ($p = 0.21$), states that confirm an origin, a destination and a departure date ($p = 0.14$), and states that confirm a departure date ($p = 0.07$).

On the other hand, in DM^{*e} model the most probable transitions lead from this user state that asks for departure timetables to the following system states: confirmation of a destination ($p = 0.16$), answer of departure timetables ($p = 0.14$), confirmation of an origin ($p = 0.13$), question of a destination ($p = 0.11$), confirmation of an origin and a destination ($p = 0.10$), confirmation of a departure date ($p = 0.08$), question of a departure date ($p = 0.06$) and question of an origin ($p = 0.04$).

Comparing the different probabilities of the available transitions of the two models, we can appreciate significant changes in the dialog strategy. The probability of providing the departure timetables immediately after they are requested has decreased dramatically (from 0.47 to 0.14) because the new model has been adapted to work with erroneous inputs, so it is better to confirm or ask for the values of some attributes before making the database query. Overall, the probabilities of the transitions to confirmation states have augmented, although in some cases they have decreased. For instance, the probability of confirming an origin and a destination simultaneously is lower in the new model (it has changed from 0.21 to 0.10). However, in DM^{*e}, the transitions to states of confirmation of an origin or a destination, separately, have meaningful probabilities, whereas in DM these transitions are very improbable (in the case of confirmation of a destination, it has changed from 0.01 in DM to 0.16 in DM^{*e}, and, in the case of confirmation of an origin, it has changed from 0.01 to 0.13). In addition, DM^{*e} incorporates transitions to states that ask for an origin or a destination, and DM does not include them. Obviously, DM^{*e} provides better alternatives for confirming or asking for a specific attribute whose value were unreliable or unknown in the curse of the dialog.

In addition, in the third training, we have adapted the initial model, DM, running 2000 simulations with errors in their inputs and allowing a purely stochastic behavior of the dialog manager (by means of not applying the pruning rules described in Section 3.2.2). Let DM^{*f} be the model obtained after this training. Table 6 shows some statistics of the DM^{*f} training. As we present in the following Section 5.2, the test of this new model has proven that the simulation technique can be used to improve the dialog manager using a purely stochastic dialog strategy.

Finally, in the fourth training, we have studied the possibilities of improving a model that is worse than the initial model, DM. This new initial model, that we called DM^{p}, has the same states and transitions of DM but we have established that all the transitions from each state have the same probability (i.e., without a priori information), and, therefore, it does not provide an acceptable dialog strategy. Thus, it seems interesting to explore the possibilities of the simulation technique to automatically learn a reasonable dialog strategy from such a model. We have adapted this model with equiprobable transitions among all the states, DM^{p}, running

Table 6
Some statistics of the DM*f training

DM*f training	Total	%	Average per dialog
Dialogs	2000		
Successful dialogs	1394	69.70	
Dialogs without errors	1175	58.75	
Dialogs with errors	825	41.25	
Errors in ORIGIN attribute	477		0.24
Errors in DESTINATION attribute	463		0.23
Errors in DEPARTURE-DATE attribute	400		0.20
System turns	9853		4.93
Answer system turns	7028		3.51
Question and/or confirmation system turns	2809		1.40
Other system turns	16		≅ 0.00
Ordinary transitions to user states	9848	99.95	
Extraordinary transitions to user states	5	0.05	
Ordinary transitions to system states	9853	100.00	
Extraordinary transitions to system states	0	0.00	

3000 simulations with errors in their inputs and allowing a purely stochastic behavior of the dialog manager (by means of not applying the pruning rules). Let DM*q be the model obtained after this training. Table 7 shows some statistics of the DM*q training. The test of this last model has allowed us to judge the utility of the simulation technique in order to improve the dialog manager using a purely stochastic dialog strategy and an initial model that provides a very inadequate dialog strategy.

5.2. Evaluation of the new models

We tested and compared the four models. In the first evaluation (Table 8), we evaluated DM and DM*, running 2000 dialogs using each model. Without introducing errors, the success dialog rate (calculated following the algorithm of Fig. 9) was close to 100% in both cases. The dialog duration decreases slightly when using the new model, DM*, due to a clear reduction of the transitions to other system states (waiting, not understanding, etc.). Moreover, the dialog manager needs to make extraordinary transitions fewer times using DM* than using DM.

In the second evaluation (Table 9), we ran 2000 dialogs using the same two models, DM and DM*, and introducing errors in the input frames. These errors consisted of changes in the values of the origin, destination

Table 7
Some statistics of the DM*q training

DM*q training	Total	%	Average per dialog
Dialogs	3000		
Successful dialogs	2074	69.13	
Dialogs without errors	1966	65.53	
Dialogs with errors	1034	34.47	
Errors in ORIGIN attribute	586		0.20
Errors in DESTINATION attribute	569		0.19
Errors in DEPARTURE-DATE attribute	493		0.16
System turns	17,269		5.76
Answer system turns	11,996		4.00
Question and/or confirmation system turns	5244		1.75
Other system turns	29		0.01
Ordinary transitions to user states	17,264	99.97	
Extraordinary transitions to user states	5	0.03	
Ordinary transitions to system states	17,269	100.00	
Extraordinary transitions to system states	0	0.00	

Table 8
Evaluation of DM and DM* without input errors

	DM	DM*
Number of dialogs	2000	2000
Success rate	99.45%	99.10%
Number of errors per dialog	0	0
System turns per dialog	4.19	4.11
Number of system turns	8377	8213
Number of answer system turns	4806	4788
Number of question and/or confirmation system turns	3426	3382
Number of other system turns	145	43
Number of extraordinary transitions (using DM-aux)	97	4

Table 9
Evaluation of DM and DM* with input errors

	DM	DM*
Number of dialogs	2000	2000
Success rate	71.25%	71.50%
Number of errors per dialog	1.12	1.12
System turns per dialog	4.49	4.36
Number of system turns	8974	8726
Number of answer system turns	4804	4782
Number of question and/or confirmation system turns	4007	3911
Number of other system turns	163	33
Number of extraordinary transitions (using DM-aux)	88	5

and departure date attributes, that is, the attributes whose correction determines the success of the dialog. Table 9 shows that the success rate goes down to nearly 71% and the duration of the dialogs goes up slightly in both models. Again, using DM*, the dialog manager selects the other system states fewer times and makes very few extraordinary transitions.

The results of this evaluation encouraged us to train and test the other model, DM*e, introducing errors in its training. Thus, we were able to compare the three models, DM, DM* and DM*e, by means of a new set of simulations. Table 10 shows the results.

We can appreciate a certain superiority of the new model, DM*e. Using this model we achieve a normalized success rate that is 0.17% better than using DM* and 1.89% than using DM. The average duration of the dialogs is 0.12 turns lower than using DM because the dialog manager barely selects the other system states.

Table 10
Evaluation of DM, DM* and DM*e with input errors

	DM	DM*	DM*e
Number of dialogs	2000	2000	2000
Success rate	71.95%	72.10%	72.60%
Number of errors per dialog	1.09	1.11	1.10
Rate of dialogs with errors	53.10%	54.10%	53.30%
Success rate (in dialogs with errors)	47.18%	48.43%	48.59%
Number of errors per dialog (in dialogs with errors)	2.04	2.06	2.06
Normalized success rate (in dialogs with errors)	**47.18%**	**48.90%**	**49.07%**
System turns per dialog	4.46	4.38	4.34
Number of system turns	8925	8756	8672
Number of answer system turns	4803	4776	4789
Number of question and/or confirmation system turns	3970	3946	3866
Number of other system turns	152	34	17
Number of extraordinary transitions (using DM-aux)	87	5	7

As in previous evaluations, the dialog manager scarcely makes extraordinary transitions when it uses the new models.

We include two examples in Figs. 11 and 12 in order to illustrate the working of the dialog system in simulation mode and also some differences depending on the use of DM and DM*e. Both examples are simulations using the same scenario (number 6 in Table 3). In the example in Fig. 11, the user and system dialog managers used the DM. In the example in Fig. 12, the dialog managers used the DM*e.

In these figures, we have included the sentences of the simulated user turns (indicated by Ux), in English (due to the extension of the figures, we have omitted the original Spanish sentences); their translations to frames, which are made by the understanding module; and also the system frames (indicated by Sx, generated by the dialog manager) and their corresponding English sentences. In the frames, the concept identifiers are followed by their confidence scores, and the attribute identifiers are followed by their values and confidence scores.

In Fig. 11, the first transition made by the system dialog manager, using the DM, is to an inadequate state of implicit confirmation of an arrival time plus waiting (S1 turn). Besides, this system decision is especially wrong because the origin and destination attributes are not implicitly confirmed (due to their high confidence scores) and an error has modified the value of the destination attribute. Then, in the S2 turn, the system

U1: I want to know departure and arrival timetables from Albacete to Madrid.
(ARRIVAL-HOUR) 0.50
  ORIGIN: Albacete 0.86
  DESTINATION: Mérida 0.86
 (DEPARTURE-HOUR) 0.35
  ORIGIN: Albacete 0.86
  DESTINATION: Mérida 0.86
S1:
(CONFIRMATION) 1.00
  ARRIVAL-HOUR: ??? 0.00
  ORIGIN-ACTUAL: Albacete 0.86
  DESTINATION-ACTUAL: Mérida 0.86
 (WAITING) 1.00
I am consulting arrival timetables. Wait a moment, please.
U2: I want to know departure and arrival timetables and prices.
(ARRIVAL-HOUR) 0.50
S2:
(NOT-UNDERSTOOD) 1.00
(NEW-QUERY) 1.00
I'm sorry but I don't understand you. Do you want anything else?
U3: I want to know departure timetables.
(DEPARTURE-HOUR) 0.36
S3:
(CONFIRMATION) 1.00
  DEPARTURE-DATE: 24-02-2006 0.00
Do you want to travel on February the twenty-fourth?
U4: No. I want to travel on July the eighth.
(REJECTION) 0.58
() 0.43
  DEPARTURE-DATE: 08-07-???? 0.43
S4:
(NOT-UNDERSTOOD) 1.00
(QUESTION) 1.00
  DEPARTURE-DATE: 08-07-???? 0.47
I'm sorry but I don't understand you. On what day do you want to leave?
U5: I want to travel on July the eighth.
() 0.43
  DEPARTURE-DATE: 08-07-???? 0.43
S5:
(CONFIRMATION) 1.00
  DEPARTURE-DATE: 08-07-???? 0.47
Do you want to leave on July the eighth?
U6: Yes. I want to travel on July the eighth.
(AFFIRMATION) 0.43
() 0.95
  DEPARTURE-DATE: 08-07-???? 0.95
S6:
(CONFIRMATION) 1.00
  DEPARTURE-HOUR: ??? 0.00
(WAITING) 1.00
(INFORMATION) 1.00
  NUMBER-OF-TRAINS: 5 0.99
  DEPARTURE-HOUR: 05.06 0.99
(NEW-QUERY) 1.00
There are 5 trains and I inform about the first one. I am consulting departure timetables. Wait a moment, please. The train leaves at 05.06. Do you want anything else?
U7: I want to know arrival timetables and prices.
(ARRIVAL-HOUR) 0.50
S7:
(WAITING) 1.00
(INFORMATION) 1.00
  ARRIVAL-HOUR: 07.06 0.99
 (NEW-QUERY) 1.00
Wait a moment, please. The train arrives at 07.06. Do you want anything else?
U8: I want to know prices.
(PRICE) 0.45
S8:
(CONFIRMATION) 1.00
  PRICE: ??? 0.00
(WAITING) 1.00
(INFORMATION) 1.00
  PRICE: 44 0.99
 (NEW-QUERY) 1.00
I am consulting prices. Wait a moment, please. The ticket costs 44 euros. Do you want anything else?
U9: No, thank you very much.
(CLOSING) 1.00

Fig. 11. Example of simulated dialog using the DM model.

U1: I want to know departure and arrival timetables from Albacete to Madrid.
(ARRIVAL-HOUR) 0.50
ORIGIN: Alicante 0.38
DESTINATION: Mérida 0.42
(DEPARTURE-HOUR) 0.34
ORIGIN: Alicante 0.38
DESTINATION: Mérida 0.42
S1:
(NOT-UNDERSTOOD) 1.00
(CONFIRMATION) 1.00
ORIGIN: Alicante 0.40
DESTINATION: Mérida 0.44
I'm sorry but I don't understand you. Do you want to go from Alicante to Mérida?
U2: No. I want to go from Albacete to Madrid.
(DEPARTURE-HOUR) 0.36
ORIGIN: Albacete 0.73
DESTINATION: Madrid 0.73
S2:
(CONFIRMATION) 1.00
DEPARTURE-DATE: 24-02-2006 0.11
ORIGIN-ACTUAL: Albacete 0.73
DESTINATION-ACTUAL: Madrid 0.73
From Albacete to Madrid, do you want to travel on February the twenty-fourth?
U3: No. I want to travel on July the eighth.
(REJECTION) 0.58
() 0.78
DEPARTURE-DATE: 08-07-???? 0.78
S3:
(INFORMATION) 1.00
NUMBER-OF-TRAINS: 5 0.99
DEPARTURE-HOUR: 05.06 0.99
DEPARTURE-DATE-ACTUAL: 08-07-???? 0.78
(NEW-QUERY) 1.00
On July the eighth, there are 5 trains and I inform about the first one. The train leaves at 05.06. Do you want anything else?
U4: I want to know arrival timetables and prices.
(ARRIVAL-HOUR) 0.50
S4:
(WAITING) 1.00
(INFORMATION) 1.00
ARRIVAL-HOUR: 07.06 0.99
(NEW-QUERY) 1.00
Wait a moment, please. The train arrives at 07.06. Do you want anything else?
U5: I want to know prices.
(PRICE) 0.45
S5:
(CONFIRMATION) 1.00
PRICE: ??? 0.00
(WAITING) 1.00
(INFORMATION) 1.00
PRICE: 44 0.99
(NEW-QUERY) 1.00
I am consulting prices. Wait a moment, please. The ticket costs 44 euros. Do you want anything else?
U6:
No, thank you very much.
(CLOSING) 1.00

Fig. 12. Example of simulated dialog using the DM^{*e} model.

transits to an even worse state of not understanding and asking for a new query. At this point, a human user would consider abandoning but the user simulator goes on. The system needs three turns (from S3 to S5) to ask and confirm the departure date. Finally, it requires another three turns to provide the requested information (departure and arrival timetables and prices).

In Fig. 12, the first transition made by the system dialog manager, using the DM^{*e}, is to a very convenient state of explicit confirmation of origin and destination attributes (S1 turn). In this way, the user simulator has the opportunity to correct both attributes (certainly, the lower confidence scores of these attributes also favour the right selection of this transition). Once the origin and destination attributes have been correctly set, the next system decision is to confirm the departure date and, this time, the system does it in only one turn. Later, it uses three turns to provide the requested information just as in the previous example. The dialog using the DM^{*e} is shorter than the dialog using the DM. It does not contain inadequate system answers and ends satisfactorily (because all the errors are detected and corrected).

The following step in our evaluation was to test the models in simulations with higher error rates. In Table 11 we compare the results obtained using DM and DM^{*e}. In the two left columns of Table 11 (labeled as *standard collaborative strategy*), we can observe that both models fail to provide adequate strategies to the dialog manager. When two serious errors are introduced in each dialog, the success rate goes down to nearly 20%.

We considered the effect of the user attitude in the development of the dialogs. A more cooperative behavior by the users could help to detect and correct a higher error rate of the system. Thus, we modified the user simulator strategy to increase its cooperation. Up to this point, the user simulator answered system questions and explicit confirmations adequately (following the rules detailed in Fig. 7), but it did not consider the data given in the system implicit confirmations. Sometimes, the user simulator provided the values of certain attributes "spontaneously". If some of these attributes were the object of a previous implicit confirmation, the system could either confirm or correct them. However, this fact happened by chance. Thus, we modified the user simulator strategy, allowing it to verify and correct the data in the implicit confirmations. Once this change

Table 11
Evaluation of DM and DM*e with higher error rates

	Standard collaborative strategy		Intensive collaborative strategy	
	DM	DM*e	DM	DM*e
Number of dialogs	2000	2000	2000	2000
Success rate	21.40%	22.10%	90.95%	93.95%
Number of errors per dialog	2.08	2.04	2.11	2.09
Rate of dialogs with errors	97.50%	96.95%	97.30%	97.50%
Success rate (in dialogs with errors)	20.86%	21.43%	88.25%	91.45%
Number of errors per dialog (in dialogs with errors)	2.03	1.98	2.16	2.14
Normalized success rate (in dialogs with errors)	**20.86%**	**20.90%**	**88.25%**	**90.60%**
System turns per dialog	3.82	3.62	3.94	3.70
Number of system turns	7649	7235	7882	7407
Number of answer system turns	4724	4761	4735	4753
Number of question and/or confirmation system turns	2683	2440	2865	2631
Number of other system turns	242	34	282	23
Number of extraordinary transitions (using DM-aux)	314	82	395	109

was implemented, we repeated the last evaluation. The two right columns of Table 11 (labeled as *intensive collaborative strategy*) show the new results. The dialog system using DM*e achieves a normalized success rate that is 2.35% better than using DM.

The last step in our evaluation was to test the models acquiring new simulations where the dialog manager worked in a purely stochastic manner (because none transition in the dialog model was pruned by heuristic rules). The user simulator worked applying the standard collaborative strategy in these acquisitions. The following tables show the results in these conditions depending on the model used by the dialog manager: DM or DM*f (in Table 12); DMp or DM*q (in Table 13).

Considering the values of Table 12, we can conclude that the simulation technique is also useful to enhancing the dialog model in this way of working of the dialog manager, and the improvement of the success rate using DM*f (from 66.70% to 69.90%, that is, an increasing of 3.20%) is better than the improvement achieved using DM*e. In addition, the improvement of the success rate in dialogs with errors (from 23.18% to 31.98%, that is, an increasing of 8.80%) is even better. The cost is just a higher length of the dialog (0.47 turns longer using DM*f).

We can also compare the results of Tables 10 and 12 to valuate the effects of the two dialog strategies: the hybrid strategy (with model transitions pruned taking into account the heuristic rules) and the purely stochastic strategy (without pruning any transition). Using the hybrid strategy and DM*e, we have achieved a success rate (in all the dialogs) of 72.60% and a success rate (in dialogs with errors) of 48.59% (see Table 10). Using the

Table 12
Evaluation of DM and DM*f

	DM	DM*f
Number of dialogs	2000	2000
Success rate	66.70%	69.90%
Number of errors per dialog	0.71	0.72
Rate of dialogs with errors	43.35%	44.25%
Success rate (in dialogs with errors)	23.18%	31.98%
Number of errors per dialog (in dialogs with errors)	1.64	1.64
System turns per dialog	4.50	4.97
Number of system turns	8993	9939
Number of answer system turns	6364	7121
Number of question and/or confirmation system turns	2586	2809
Number of other system turns	43	9

Table 13
Evaluation of DM^{p} and DM^{*q}

	DM^{p}	DM^{*q}
Number of dialogs	2000	2000
Success rate	35.55%	68.75%
Number of errors per dialog	0.55	0.56
Rate of dialogs with errors	34.25%	36.15%
System turns per dialog	9.03	5.60
Number of system turns	18,069	11,206
Number of answer system turns	12,828	7861
Number of question and/or confirmation system turns	5173	3323
Number of other system turns	68	22
Incoherent turns per dialog	3.98	1.32
Incoherent turns per turn	0.44	0.24

purely stochastic strategy and DM^{*f}, we have achieved a success rate (in all the dialogs) of 69.90% and a success rate (in dialogs with errors) of 31.98% (see Table 12). Thus, the increasing of failure is abrupt enough in the dialogs where errors have been introduced. In these cases, the purely stochastic strategy is worse because transitions to system answers are chosen more times (including many situations where attributes have unreliable values and it would be better to confirm or ask for them). Using the hybrid strategy, the dialog manager chooses 55.22% of answers and 44.58% of questions and confirmations. Using the purely stochastic strategy, the dialog manager chooses 71.65% of answers and 28.26% of questions and confirmations.

In addition, considering the results shown in Table 13, we can also conclude that the simulation technique provides an automatic procedure for obtaining a reasonable dialog model, as DM^{*q}, from an initial model as DM^{p}, which is very unsatisfactory. The improvement of the success rate using DM^{*q} is 33.20%, and the shorter duration of the dialogs is the more important reason of this enhancement. The length of the dialog has been also significantly reduced in 3.43 turns (i.e., a reduction of 37.98%), and this fact demonstrates that the new model facilitates better ways to satisfy the users requests. In this table, we also report the average number of incoherent system turns per dialog (i.e., system answers that would be considered unacceptable by real users as, for instance, to provide prices when the user asks for timetables). The number of incoherent system turns has been reduced in 2.66 turns per dialog (the probability of generation of an incoherent system turn has decreased from 0.44 to 0.24). Thus, the simulation technique has allowed us to obtain DM^{*q}, a dialog model that provides a better dialog strategy in both quantitative and qualitative terms.

Finally, we acquired a set of dialogs with human users. Although it is rather a preliminary evaluation, the results are promising and they encourage us to continue using and enhancing the methodology presented in this paper.

We required the cooperation of 8 expert users (colleagues of our research group). Each user acquired 2 dialogs for each one of the 14 scenarios shown in Table 3. One dialog was acquired with the dialog manager using DM (the initial BASURDE model) and the other dialog was acquired using DM^{*e} (the modified model after the automatic strategy learning by simulations). Thus, we acquired a total of 224 dialogs, one half using DM and the other half using DM^{*e}.

The users acquired the dialogs through a textual interface, they could read the natural language sentences generated by the dialog system (and nothing else) and wrote the sentences of their turns. The dialog system simulated errors in the values of origin, destination and departure date attributes, in the understanding module. We acquired the dialogs using this procedure, instead of using a speech interface, in order to test dialog situations similar to those considered in the training of the new model.

In the 112 dialogs acquired using DM, with an average of 0.67 errors per dialog, we achieved a success dialog rate of 75.9%. On the other hand, in the 112 dialogs acquired using DM^{*e}, with an average of 0.70 errors per dialog, we achieved a success dialog rate of 83.9%. The use of the new model increased the success dialog rate by 8%. In addition, it also reduced the duration of the dialog and the occurrence of transitions to system states that were inadequate (states of waiting, not understanding and others).

6. Conclusions

In this paper, we have presented an overview of a new methodology of user simulation applied to the evaluation and refinement of dialog systems. Our research in the field of dialog systems is being developed within the statistical approach. Usual weaknesses of the stochastic dialog systems are the limitations of the training corpus and the high cost of an evaluation carried out with the collaboration of real users. We have considered the user simulation technique as an alternative way of testing and improving our dialog system.

Our set of user simulator modules has been designed incorporating statistical information (the UDM works with the same stochastic model trained from the BASURDE corpus and chooses the following system states according to the probabilities of the transitions in that model) and heuristic information (the UDM determines the following user states taking into account the dialog history, the goals of the scenario, and a collaborative strategy). These modules have been integrated into the dialog system allowing us to execute series of thousands of dialogs, without cost and at very high speed, in order to make the dynamic adaptation of the stochastic dialog models (automatic strategy learning) and the comparative test of the dialog system working with different dialog models.

The automatic strategy learning of the stochastic dialog models has become a reality by means of this user simulation technique. The evaluation shows a slight but clear improvement of the dialog system when it uses the new models. We have obtained better success rates (around 1% or 2% of improvement), shorter dialogs, fewer transitions to the system states that are inadequate for the natural progress of the dialogs, and very few calls to the extraordinary procedures of searching for transitions (that is, the new models have almost always provided one or more reasonable transitions to continue the dialogs). In addition, we have successfully applied the same simulation technique in order to enhancing the dialog manager working in a purely stochastic way (that is, not applying heuristic rules in the decision of choosing the system behavior). Given that in these conditions the performance of the dialog manager is worse, there are more possibilities for improving and we have achieved more clear benefits (around 3% of improvement in total success rate, and around 8% of improvement in success rate in dialogs with errors).

Summarizing, the user simulator has allowed us to acquire useful synthetic dialogs, which conclude satisfactorily (achieving dialog goals) in a reasonable time (duration of dialog below a given threshold), in different situations: (a) when the dialog manager works using the recommended hybrid dialog strategy; (b) when the dialog manager follows a pure stochastic dialog strategy (deactivating all the heuristic rules); and (c) even when it uses a bigram model with equiprobable transitions among all their states (which provides an initial unacceptable strategy). The proposed technique provides some degree of improvement in all these cases.

As future work, we are considering the extension of the simulated scenarios to all the scenarios defined in the DIHANA (Benedí et al., 2006) task as well as the application of the same technique to the initial model trained from the DIHANA corpus (BASURDE and DIHANA have been research projects that dealt with the same dialog task). We also would like to explore the possibilities of the user simulation technique to facilitate the adaptation of the dialog system to different tasks, which is one of the objectives of our new research project, called EDECAN (Lleida et al., 2006).

Acknowledgements

This work has been partially funded by Spanish MEC and FEDER under project TIN2005-08660-C04-02, Spain. We also thank David Griol and the anonymous reviewers for valuable comments on a draft of this paper.

References

Benedí, J.M., Lleida, E., Varona, A., Castro, M.J., Galiano, I., Justo, R., López, I., Miguel, A., 2006. Design and acquisition of a telephone spontaneous speech dialogue corpus in Spanish: DIHANA. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), Genove, Italy, pp. 1636–1639.

Bonafonte, A., Aibar, P., Castell, N., Lleida, E., Mariño, J., Sanchis, E., Torres, M., 2000. Desarrollo de un sistema de diálogo oral en dominios restringidos. Actas de las I Jornadas en Tecnología del Habla, Sevilla, Spain.

Eckert, W., Levin, E., Pieraccini, R., 1997. User modeling for spoken dialogue system evaluation. In: Proceedings of the ASRU – IEEE Workshop, Santa Barbara, USA.

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

Lleida, E., Segarra, E., Torres, M.I., Macías, J., 2006. EDECÁN: sistema de diálogo multidominio con adaptación al contexto acústico y de aplicación. Actas de las IV Jornadas en Tecnología del Habla, Zaragoza, Spain, pp. 291–296.

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

Martínez, C., Sanchis, E., García, F., Aibar, P., 2002. A labeling proposal to annotate dialogues. In: Proceedings of the 3rd LREC, pp. 1577–1582.

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

Schatzmann, J., Georgila, K., Young, S., 2005. Quantitative evaluation of user simulation techniques for spoken dialog systems. In: Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue, Lisboa, Portugal.

Scheffler, K., Young, S., 2001. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In: Proceedings of the NAACL, Workshop on Adaptation in Dialogue Systems, Pittsburgh, USA, pp. 64–70.

Torres, F., Sanchis, E., Segarra, E., 2003. Development of a stochastic dialog manager driven by semantics. In: Proceedings of the 8th Eurospeech, Geneva, Switzerland, pp. 605–608.

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

Torres, F., Sanchis, E., Segarra, E., 2005b. Learning of stochastic dialog models through a dialog simulation technique. In: Proceedings of the 9th Eurospeech, Lisboa, Portugal, pp. 817–820.

Young, S., 2002. The statistical approach to the design of spoken dialogue systems, Cambridge University, Technical Report CUED/F-INFENG/TR.433, September 2002.