ELSEVIER

Speech Communication 40 (2003) 387–407

SPEECH COMMUNICATION

www.elsevier.com/locate/specom

Assessment of dialogue systems by means of a new simulation technique

R. López-Cózar *, A. De la Torre, J.C. Segura, A.J. Rubio

Dpto. Electrónica y Tecnología de Computadores, Universidad de Granada, 18071 Granada, Spain

Received 24 August 2001; accepted 27 June 2002

Abstract

In recent years, a question of great interest has been the development of tools and techniques to facilitate the evaluation of dialogue systems. The latter can be evaluated from various points of view, such as recognition and understanding rates, dialogue naturalness and robustness against recognition errors. Evaluation usually requires compiling a large corpus of words and sentences uttered by users, relevant to the application domain the system is designed for. This paper proposes a new technique that makes it possible to reuse such a corpus for the evaluation and to check the performance of the system when different dialogue strategies are used. The technique is based on the automatic generation of conversations between the dialogue system, together with an additional dialogue system called *user simulator* that represents the user's interaction with the dialogue system. The technique has been applied to evaluate a dialogue system developed in our lab using two different recognition front-ends and two different dialogue strategies to handle user confirmations. The experiments show that the prompt-dependent recognition front-end achieves better results, but that this front-end is appropriate only if users limit their utterances to those related to the current system prompt. The prompt-independent front-end achieves inferior results, but enables front-end users to utter any permitted utterance at any time, irrespective of the system prompt. In consequence, this front-end may allow a more natural and comfortable interaction. The experiments also show that the re-prompting confirmation strategy enhances system performance for both recognition front-ends.

Keywords: Dialogue systems; Speech recognition; Speech understanding; Dialogue management; System evaluation

1. Introduction

Dialogue systems have developed greatly in the last decade, such systems being applied to airline and railway information, weather forecasts, ticket booking, automatic telephone switchboards, multimedia services, routing systems, etc. (Zue et al., 1997, 2000; Gorin et al., 1997; Kellner et al., 1997, 1998; Lamel et al., 1997, 1998; Buntschuh et al., 1998). These systems offer many advantages compared to other

* Corresponding author. Tel.: +34-958-243-271; fax: +34-958-243-230.

E-mail addresses: rlopezc@ugr.es (R. López-Cózar), atv@ugr.es (A. De la Torre), segura@ugr.es (J.C. Segura), rubio@ugr.es (A.J. Rubio).

URL: http://ceres.ugr.es/~ramon.

0167-6393/02/$ - see front matter
doi:10.1016/S0167-6393(02)00126-7

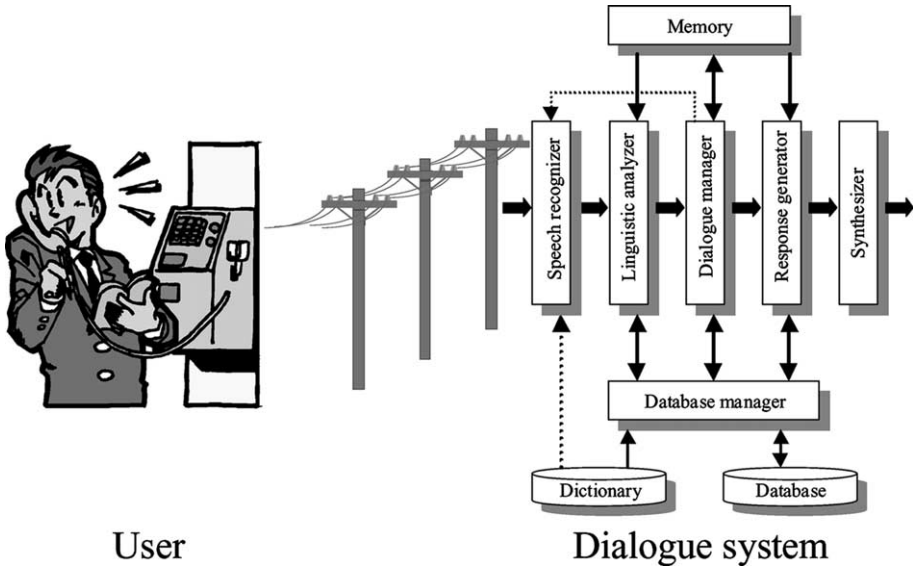

Fig. 1. A user interacting with a dialogue system using the telephone.

communication technologies, such as those based on touch-tone telephones. Users can communicate using speech and can interact intuitively, since they do not need to learn and remember complex instructions. Consequently, these systems provide an easier and more natural access to a variety of services. The structure of a generic dialogue system is shown in Fig. 1.

The speech recognizer receives the user's voice signal and generates one or more recognized sentences as output. The linguistic analyzer generates semantic representations from the recognized sentences. The dialogue manager controls the dialogue status and decides the response to be generated by the system, considering the semantic interpretation obtained and the dialogue history. The response generator builds the system response, which is finally transformed into a voice signal by a synthesizer.

In order to achieve an acceptable performance, the development of these systems requires several aspects to be considered. For example, they require the compilation, transcription and labeling of large training corpora (words, user-system interactions, etc.). This material is frequently difficult to obtain. The dialogue strategies must be carefully designed in order to create friendly interfaces, especially for inexperienced users (Rosset et al., 1999; Souvignier et al., 2000). The response time is a very important factor in the design of such systems. If a response is not rapidly generated, users may be dissatisfied. Another major issue is how to deal with problems of comprehension. Since dialogue systems can never be absolutely certain they have correctly understood the user, they must constantly verify their assumptions. The above problems, among others, suggest that new tools to facilitate the design, development and evaluation of dialogue systems are needed.

A great deal of research has been devoted to enhancing the performance of such systems in terms of word recognition and speech understanding (Rabiner and Juang, 1993; Boros and Heisterkamp, 1999; Noeth et al., 1999; Schadle et al., 1999). We believe, as do others, that a major challenge to this technology concerns the better incorporation of models of human-to-human interaction (Swerts and Ostendorf, 1997), which requires us to study the interaction between users and systems from a more general point of view, such as dialogue structure, dialogue efficiency and task achievement. As a tool to facilitate the development and testing of dialogue systems, this paper proposes a user simulator; this is an additional dialogue system that interacts automatically with the dialogue system to be tested, and represents a real user. The idea of evaluating a dialogue system by using another system that interacts with it is not a new one; for example, Levin et al. (2000) presented a simulator for learning dialogue strategies, defined as a generative stochastic

model. Given the current state of the dialogue system and a prompt, the simulator produces the semantic representation of an utterance in a template form, like a template generated by the user. The main drawback of this work is that recognition and understanding errors are not considered. Araki et al. (1997) and Araki and Doshita (1997) presented an evaluation method that relied on dialogues generated by two systems that interact automatically using text messages, in such a way that recognition and understanding errors are simulated by a module, called *coordinator*, that drops context words into the messages at a given rate. The technique proposed in the present paper is also concerned with evaluating a dialogue system by its automatic interaction with another (the simulator), but instead of receiving text messages, the dialogue system receives a user's voice as the input. This technique presents the advantage of considering real speech recognition phenomena existing in a speech-based user–system interaction.

Several studies have been presented to enhance these systems by considering the performance of some of their components in order to adapt the behavior. For example, Niimi and Nishimoto (1999) presented an analytical method that relies on the performance of the recognition sub-system to adapt the behavior of the dialogue system; specifically, recognition word error rates are used to decide the most appropriate confirmation strategy for the system. One difference between this method and ours is that the present paper considers the co-operation between several components of a dialogue system, mainly the recognizer and the linguistic analyzer, and pays more attention to speech understanding than to speech recognition. We believe the really important question is how well the systems understand the utterances generated by the users, and how their ability to understand affects the tasks to be performed. In fact, the present paper shows that some word recognition errors can be rectified by the linguistic analyzer of a dialogue system employing an implicit recovery (IR) technique. For this reason, the evaluation carried out in this paper focuses not only on word accuracy but also on sentence recognition (SR) and understanding (SU), IR and task completion (TC). We think the latter measure is the most important one, since it takes into account the co-operation among the diverse components of the system as well as the interactivity between the dialogue system and the users.

The remainder of the paper is organized as follows: the technique proposed in this paper is presented in Section 2, where we describe the corpora needed to set up a user simulator. Special attention is paid to the simulator module and its performance, including the strategies employed to handle error recovery, confirmations and the management of scenario goals. The section presents an algorithm showing how the user simulator generates its responses, and an example of a generated dialogue. Finally, a comment is made on the reusability of the corpora and the modules used to set up the simulator. Section 3 presents the experiments, in which we focus on word accuracy, SR and SU, IR and TC. The section begins with a description of the experimental set up concerning the recognition front-ends, SU and the procedure to cancel the automatic generation of dialogues. The experimental results show the performance of a dialogue system under four different lab-generated conditions created by (i) two different recognition front-ends, and (ii) two different strategies to handle user confirmations. Finally, the conclusions and some lines for future work are presented in Section 4.

2. The simulation technique

This paper proposes a technique to evaluate dialogue systems by means of automatically generated conversations. The technique provides dialogue system developers with a simple means of checking system features (word accuracy, SU, etc.) and the dialogue strategies employed to build the system. Moreover, it can be used to measure the effects of changing these strategies. The technique is based on the use of an additional dialogue system, called *user simulator* that represents the user interacting with the dialogue system. Fig. 2 shows the interconnection between a user simulator and the SAPLEN dialogue system, developed in our lab to deal with the telephoned product orders and queries made by clients of fast food restaurants in Spain (López-Cózar et al., 1998, 1999, 2000a,b).

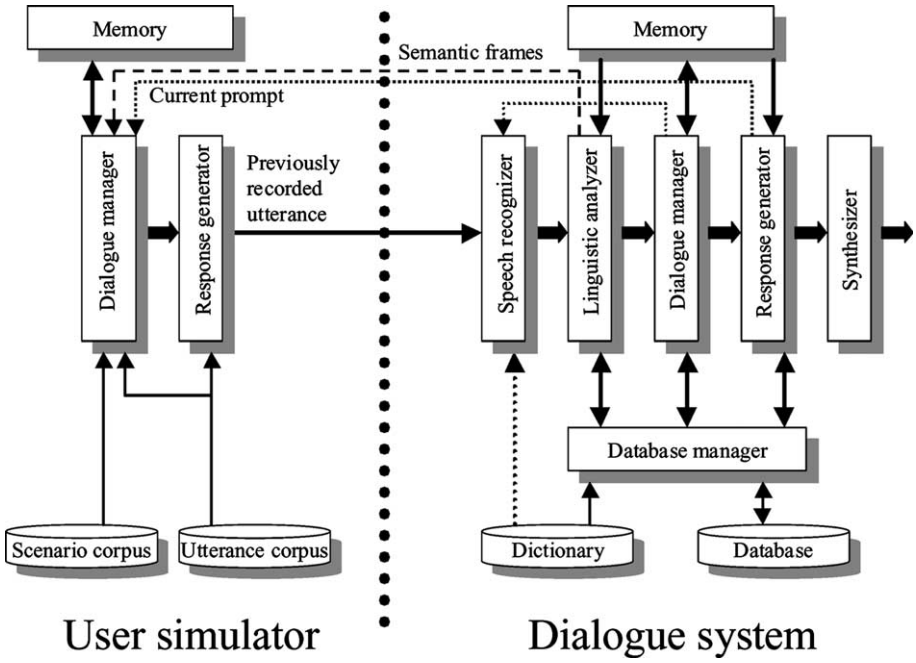

Fig. 2. Interconnection between a simulator and the SAPLEN dialogue system.

2.1. Scenario corpus

This corpus represents the goals of users interacting with the dialogue system. A wide range of scenarios must be carefully designed to consider a representative sample of possible user goals. We created 18 different scenarios for the experiments, all related to the fast food domain; each scenario contained the following elements (goals): five product orders (food and/or drinks) plus the user's telephone number, post code and address. An example of a scenario used in the experiments is shown (Fig. 3).

For every interaction, the simulator uses a particular scenario specified by the system developer at the start of the dialogue generation, and attempts to achieve the specified goals, i.e. the simulator tries to make the dialogue system understand all the goals in the scenario. The system developer also specifies at the start of the interaction how many dialogues the simulator must generate using the scenario. The goals are represented semantically in the scenarios by means of semantic frames (Allen, 1995). Thus, to create the scenario corpus for the experiments, we used the 90 semantic representations corresponding to the 90 sentences (a)–(e) in text format shown in Table 1. As explained in Section 2.2, these sentences were selected from a dialogue corpus previously obtained in a fast food restaurant (different speakers read them to create the utterance corpus). Thus, a sentence has one semantic representation and several utterances associated with it.

The table sets out the eight sentence types considered, the number of sentences for each type, the number of utterances corresponding to the sentences to be recorded by each speaker, and whether the sentences are explicitly represented as user goals in the scenarios created to evaluate the system. The 15 semantic representations regarding sentences (f)–(h) were also used in the experiments. In particular, the simulator uses semantic representations (f) and (g) to answer the prompts generated by the dialogue system to confirm data items, and also semantic representations (h) to generate responses that initiate correction sub-dialogues. As these responses may be generated at any time during a conversation, irrespective of the current scenario, semantic representations (f)–(h) are not explicitly included in scenarios to represent user goals.

```
# the user wants to order a ham sandwich

AMOUNT="1"
FOOD="SANDWICH"
TYPE="HAM"

# the user wants to order a cheese sandwich

AMOUNT="1"
FOOD="SANDWICH"
TYPE="CHEESE"

# the user wants to order a large beer

AMOUNT="1"
DRINK="BEER"
SIZE="LARGE"

# the user wants to order a chocolate milkshake

AMOUNT="1"
DRINK="MILKSHAKE"
TASTE="CHOCOLATE"

# the user wants a vegetable and turkey salad

AMOUNT="1"
FOOD="SALAD"
TYPE="VEGETABLE TURKEY"

# the user's telephone number

TELEPHONE_NUMBER="958275360"

# the user's post code

POST_CODE="18001"

# the user's address

ID_ADDRESS="STREET"
NAME="ANDALUCIA"
BUILDING_NUMBER="58"
FLOOR="FIRST"
LETTER="E"
```

Fig. 3. A sample scenario.

Table 1
Sentence selection to create the utterance corpus

Sentence type	Different sentences selected in the dialogue corpus	Utterances to record corresponding to the sentences	Scenario goal
(a) Food order	21	63	Yes
(b) Drink order	15	60	Yes
(c) Phone number	18	18	Yes
(d) Post code	18	18	Yes
(e) Address	18	90	Yes
(f) Affirmative confirmation	5	5	No
(g) Negative confirmation	5	5	No
(h) Error indication	5	5	No
Total	105	264	

2.2. Utterance corpus

This corpus contains utterances concerning the dialogue system application domain recorded by several speakers. This corpus is analyzed and tagged, so that the user simulator can use these utterances to generate its responses. The tagging used is only concerned with establishing the correspondences between sentences and utterances, focusing on the file names: the phonetic transcription of a sentence is a text file whose name refers to the sentence type (for example, food_order_1.txt); its correct semantic representation is a file that contains the tag "semantic representationcor" in its name (for example, food_order_1.SRcor); and the utterances the speakers record when reading the sentence are files whose name includes the sentence type, the speaker, the recording date and time and the tag "wav" (for example, food_order_1_ram_100413184349.wav). The phonetic transcriptions are used to automatically create the conversation log files that are used to evaluate the system.

When the simulator needs an utterance to generate a response, it selects one whose semantic representation is in the scenario representing a user goal. Thus, focusing on the file names, the procedure explained above for the tagging operation makes it very easy to search for an utterance associated with a semantic representation. Furthermore, as different utterances in the corpus may share the same semantic representation, a variety of possible utterances can be used at a given moment of a conversation to generate a response. For example, if the user simulator needs an utterance whose semantic representation is "affirmative confirmation", then it can choose any of the following utterances: "yes", "yes, you are right", "oh yes", etc., as the three utterances have the same semantic representation. Additionally, since utterances recorded by several speakers can be used during the generation of a conversation to answer a system prompt, the simulation technique can test whether the dialogue system correctly handles the voice and speaking style of different users.

In order to set up the utterance corpus for the experiments, we used a dialogue corpus that contained 523 dialogues, recorded in a fast food restaurant, between clients and the restaurant staff (López-Cózar et al., 1998); the clients did not know they were being recorded. Each dialogue represents the interaction between a client and an employee. The clients typically order products (food and drinks) and ask questions regarding prices, sizes, ingredients, etc. The recording quality is very poor since the restaurant is a very noisy environment, with sounds such as music, cash machines, doors, chairs, other voices, etc. Analysis of this corpus informs us about: (i) vocabulary for the dialogue system, (ii) spontaneous phenomena in clients' utterances—for example, many clients begin utterances before finally deciding what they are going to order, and use fillers like "uh", (iii) type and structure of sentences generally uttered by clients, and (iv) responses the dialogue system must generate to interact with clients.

From this corpus, 105 sentences (product orders, telephone numbers, post codes, addresses, affirmative confirmations, negative confirmations and error indications) were randomly selected (see Table 1). All the sentences—except the confirmations and the error indications—were selected to be phonetically and semantically different. When we selected them, if a new sentence was the same, from both points of view, as a previously selected one, then a new sentence was randomly selected. That is, the 21 food orders, 15 drink orders, 18 phone numbers, 18 post codes and 18 addresses are phonetically and semantically different. The confirmations and the error indications were randomly selected to be only phonetically different (for example, "yes", "yes, you are right", "yeah", "oh yes", etc.). The five affirmative confirmations used share the same semantic representation "(TYPE = AFF_CONF)", the five negative confirmations share the same SR "(TYPE = NEG_CONF)" and the five error indications share the same semantic representation "(TYPE = ERROR)".

Besides recording the sentences that are considered to be user goals for inclusion in scenarios, it is also necessary to record utterances to recover from errors that may occur in recognizing these sentences. This means that it is necessary to record two additional utterances per food order, corresponding to *amount* and *type* data items; three additional utterances per drink order, corresponding to *amount*, *size* and *taste* data items; and four additional utterances per address, corresponding to *name*, *building number*, *floor* and *letter* data items, according to the typical format of Spanish addresses. It is also necessary to record *confirmation* and *error indication* utterances for the dialogue management. The simulator uses the confirmation utterances to answer the prompts generated by the dialogue system, to confirm data items, and it uses the error indication utterances to initiate recovery sub-dialogues. These sentences are not represented as goals in the scenarios since they are scenario-independent, i.e. the user simulator uses them in all the dialogues, depending on the conversation status and the system prompt.

The simulator can use a total of 264 different sentences to interact with the dialogue system. Nine speakers spontaneously read these sentences to create the utterance corpus, using fillers (such as *uh*) if they wished: four of them spoke standard Spanish; four spoke Spanish from southern Spain; and one speaker was a Japanese woman who spoke Spanish. All but the Japanese woman were lab colleagues experienced in the development of dialogue systems. The utterances were recorded under quiet lab conditions using a

personal computer (PC) and stored using 16 bits per sample at 8 kHz. The semantic representations corresponding to the sentences were automatically created using the linguistic analyzer of the SAPLEN system, which creates one frame for every input sentence. Thus, in all, the utterance corpus contained 2376 utterances (recorded sentences), 264 phonetic transcriptions (text files) and 264 semantic representations (files containing a semantic frame).

For some application domains, compiling human-to-human dialogues to create the utterance corpus could be a very difficult task. For such domains, it might be possible to take into account expert knowledge about the type and structure of the sentences that users may utter to speak to the dialogue system, and then use these sentences to create the utterance corpus. To verify that the corpus is large enough, the dialogue system developer should also use his/her expert knowledge concerning the system performance to check that the corpus contains all the necessary utterances for the interaction, e.g., in addition to utterances representing user goals, the corpus must contain the utterances needed for dialogue management (confirmations, error indications and data items in the goals). This condition represents the lower limit for the corpus size in the sense that it allows an interaction when at least one scenario is used. Taking into account the complexity of the application domain, the developer should decide how many utterances representing different user goals must be recorded in order to create different scenarios to evaluate and check the dialogue system. In the present experiments, only 18 different scenarios were designed to present the proposed simulation technique, but more scenarios should be employed to obtain more accurate experimental results.

2.3. Dialogue manager, response generator and memory modules

The dialogue manager handles the interaction with the dialogue system using 31 *if-then-else* rules to process the dialogue system prompts, together with 28 procedures to deal with scenario goals, semantic representations and dialogue history. It decides the simulator's next response after receiving an input, i.e. the current prompt of the dialogue system and the semantic representation (frame) obtained from analysis of the simulator's previous response. The dialogue manager also considers the goals in the current scenario, the correct semantic representation of the previous utterance it generated as a response, and the dialogue history. The dialogue manager output is a semantic representation (frame) that comprises the input for the response generator. The latter takes the semantic representation provided by the dialogue manager and searches for an utterance in the corpus associated with this semantic representation. When the utterance is found, the user simulator includes the associated phonetic transcription in the log file that is automatically created to store the conversation between the user simulator and the dialogue system. The utterance is sent to the dialogue system speech recognizer that represents the user's voice. The memory module tracks the history of the conversation by storing the semantic representations obtained from the dialogue system, thus allowing the dialogue manager to know the product orders that have been correctly understood by the dialogue system at a given moment of the conversation.

2.4. Performance of the user simulator

Setting up a user simulator for a dialogue system requires taking into account all the possible prompts of the dialogue system and defining the appropriate simulator actions to generate an appropriate response for each prompt. We define the actions as *if-then-else* rules that are set up in the dialogue manager. This section explains how we set up three kinds of action: error recovery, confirmations and management of scenario goals.

Dialogue systems generally make recognition and understanding errors during interaction with users. Typically, when dialogue systems do not understand utterances, users are requested to repeat the whole sentence, or only the words that were not recognized or understood. In order to take this procedure into account, the user simulator detects the data items in the scenario goals and uses them appropriately to answer the dialogue system correction prompts. To achieve this, a procedure was established in the

simulator dialogue manager for which the input was the current semantic representation (frame) produced by the dialogue manager as the output for the response generator. This procedure detects the slots in the semantic representation and generates N possible semantic representations (N frames containing just one slot) that may be needed to answer the dialogue system correction prompts. For example, the semantic representation of the utterance "I would like to have two ham sandwiches, please" is (AMOUNT = "2", FOOD = "SANDWICH", TYPE = "HAM"), which contains three data items that are also semantic representations: (AMOUNT = "2"), (FOOD = "SANDWICH") and (TYPE = "HAM"). Then, if the dialogue system does not understand this utterance correctly and generates the prompt: "How many ham sandwiches do you want?" then the user simulator dialogue manager uses the semantic representation (AMOUNT = "2") to generate the appropriate correction response. This recovery procedure can be seen in the following dialogue, where the error in the utterance comprehension is a consequence of a word recognition deletion error.

Dialogue 1. Example of error recovery handled by the SAPLEN system:

Speaker	Utterance	Simulator action
SAPLEN:	What would you like to have?	–
SIMULATOR:	I want two ham sandwiches	Use a scenario goal
Recognizer output:	I want ham sandwiches	
SAPLEN:	How many sandwiches do you want?	–
SIMULATOR:	Two	Error recovery
Recognizer output:	Two	
SAPLEN:	Ok, two ham sandwiches. Please say your telephone number	–

In other cases, the dialogue system may understand something completely different to the user input. For example, a user may utter "I want two ham sandwiches" and the dialogue system may understand the user wants "One cheese sandwich". On the one hand, if the dialogue system uses an explicit confirmation strategy then it prompts the user to confirm the data items using a "yes"/"no" response (Bouwman and Hulstijn, 1998; Lavelle et al., 1999). This procedure can be observed in the following example of dialogue.

Dialogue 2. Example of explicit confirmation:

Speaker	Utterance
SYSTEM:	What would you like to have?
USER:	I want two ham sandwiches
Recognizer output:	I want one cheese sandwich
SYSTEM:	Did you say you want one cheese sandwich?
USER:	No
SYSTEM:	Ok, please repeat your last order
USER:	I want two ham sandwiches
Recognizer output:	I want two ham sandwiches
SYSTEM:	Did you say you want two ham sandwiches?
USER:	Yes
Recognizer output:	Yes
SYSTEM:	Ok, two ham sandwiches. Please say your telephone number

On the other hand, if the dialogue system uses an implicit confirmation strategy then the data items of the previous user utterance are mentioned in the following turn of the dialogue system, and it considers them confirmed unless the user initiates a correction sub-dialogue. This procedure can be observed in the following example of dialogue.

Dialogue 3. Example of implicit confirmation handled by the SAPLEN system:

Speaker	Utterance	Simulator action
SAPLEN:	What would you like to have?	–
SIMULATOR:	I want two ham sandwiches	Use a scenario goal
Recognizer output:	I want one cheese sandwich	
SAPLEN:	Ok, one cheese sandwich. Please say your telephone number	–
SIMULATOR:	It is wrong	Correction sub-dialogue
Recognizer output:	It is wrong	
SAPLEN:	Ok, I'll remove the cheese sandwich. Please repeat your last order	–
SIMULATOR:	I want two ham sandwiches	Use a scenario goal
Recognizer output:	I want two ham sandwiches	
SAPLEN:	Ok, two ham sandwiches. Please say your telephone number	–

The SAPLEN system uses both confirmation strategies: implicit confirmations are used for product orders to avoid a confirmation overload that might be excessive for users, and explicit confirmations are used for the user data (telephone number, post code and address), products ordered, price payable and estimated delivery time. The two confirmation strategies are easily set up by comparing the semantic representation obtained by the dialogue system (SR_{obt}) from the analysis of the user utterance with the correct semantic representation of the utterance (SR_{cor}), defined a priori. In order to set up the confirmation procedure, we considered two types of incorrect semantic representation: *partial* and *contradictory*. Suppose that a semantic representation is a compound of s_i slots to make the following definitions:

(i) SR_{obt} is considered partial with respect to its correct semantic representation $SR_{cor} = \{s'_1, s'_2, \ldots, s'_m\}$ if SR_{obt} only contains some slots s_i in SR_{cor} and does not contain any slot that is not in SR_{cor}, i.e., $SR_{obt} \subset SR_{cor}$.
(ii) SR_{obt} is considered contradictory with respect to its correct semantic representation $SR_{cor} = \{s'_1, s'_2, \ldots, s'_m\}$ if SR_{obt} contains at least one slot that is not in SR_{cor}, i.e., $SR_{obt} \not\subset SR_{cor}$.

Incorporating these definitions, Table 2 shows the actions performed by the user simulator to handle both explicit and implicit confirmations.

Table 2
Simulator actions to handle confirmations

Semantic representation comparison	Explicit confirmation strategy	Implicit confirmation strategy
$SR_{obt} = SR_{cor}$	Generate affirmative confirmation	Do not generate any confirmation
$SR_{obt} \neq SR_{cor}$	Generate negative confirmation	IF partial semantic error THEN error recovery IF contradictory semantic error THEN correction turn

The table shows that when the dialogue system uses the explicit confirmation strategy, if $SR_{obt} = SR_{cor}$ then the simulator generates an affirmative confirmation, and if $SR_{obt} \neq SR_{cor}$ then it generates a negative confirmation (shown in Dialogue 2). It can also be observed that using the implicit confirmation strategy, if $SR_{obt} = SR_{cor}$ then the user simulator does not generate any confirmation, and if $SR_{obt} \neq SR_{cor}$ then it considers the type of semantic error in order to decide the appropriate action to take. If the error is partial the simulator uses a data item to answer the following turn of the dialogue system, which is generated to rectify the error (shown in Dialogue 1). If the error is contradictory then the simulator generates a correction turn to initiate a correction sub-dialogue (shown in Dialogue 3).

The user simulator selects the scenario goals after considering the prompts generated by the dialogue system. When the automatic dialogue generation begins, the simulator analyses the current scenario to identify the goals it contains, and during the dialogue uses several rules that define the correspondences between prompt types (for example, *prompt_for_product_order*, *prompt_for_post_code*, etc.) and goal types (for example, *food_order*, *user_post_code*, etc.). The simulator analyzes the current prompt to decide its type and looks for a previously unused goal associated with the prompt type, after considering the correspondence rules. If a goal is found, the simulator marks it as *used* and the goal's semantic representation is the input for the simulator response generator. The simulator then analyzes the semantic representation to determine the data items contained and prepares responses to rectify possible partial semantic errors. If the simulator does not find an unused goal for the prompt type then it generates a negative response, as can be observed in turn (36) of the dialogue example shown in the following section.

2.5. Algorithm for response generation and example of a generated dialogue

The procedures explained in the previous section can be represented algorithmically as described below. The algorithm has two main tasks: firstly, to decide the semantic representation (SR) to generate the following response and, secondly, to choose an utterance (voice signal file) and a phonetic transcription (text file) to convey that semantic representation.

```
simulator_response_generation(Inputs: prompt, SR_obt; Output: voice_signal_file)
  /* SR initialization */
  SR = ""
  /* Part 1: implicit confirmations */
  if ((prompt ≠ explicit confirmation) and (not first simulator response)) then
        {if (SR_obt ≠ SR_cor) and (SR_obt ⊄ SR_cor) then SR = ERROR_INDICATION}
  /* Part 2: explicit confirmations */
  if (SR = "") {
    if (prompt = TELEPHONE_OK?) then {if (SR_obt = SR_cor) then SR = AFFIRMAT_CONF
                                      else SR = NEGAT_CONF}
    else
    if (prompt = POST_CODE_OK?) then {if (SR_obt = SR_cor) then SR = AFFIRMAT_CONF
                                      else SR = NEGAT_CONF }
    else
    ...
    if (prompt = ORDERS_OK?) then {check_ordered_products()
                                   if (ok) then SR = AFFIRMAT_CONF
                                   else SR = NEGAT_CONF }
  }
  /* Part 3: product-order error recovery */
```

```
if (SR = "") {
   if (prompt = AMOUNT?) then SR = AMOUNT_ITEM
   else
   if (prompt = SIZE?) then SR = SIZE_ITEM
   else
   ...
   if (prompt = TASTE?) then SR = TASTE_ITEM
}
/* Part 4: management of scenario goals */
if (SR = "") {
   if (prompt = SOMETHING_TO_EAT?) then {search_scenario_for_food_order(goal)
                                         SR = goal
                                         mark_as_used(goal)
                                         create_data_items(goal)}
   else
   if (prompt = SOMETHING_TO_DRINK?) then {search_scenario_for_drink_order(goal)
                                           SR = goal
                                           mark_as_used(goal)
                                           create_data_items(goal)}
   else
   ...
   if (prompt = ANYTHING_ELSE?) then {search_scenario_for_order(goal)
                                      if found(goal) then {
                                            SR = goal
                                            mark_as_used(goal)
                                            create_data_items(goal)}
                                      else SR = NEGAT_CONF
}
/* Part 5: response generation */
look_for_file(SR, phonetic_transcription.txt, voice_signal.wav)
include_in_log_file(prompt, phonetic_transcription.txt)
voice_signal_file = voice_signal.wav
/* Part 6: dialogue history storage */
store_in_memory (SR_obt)
/* set correct semantic representation as the desired semantic representation */
SR_cor = SR
End
```

The first part of the algorithm is concerned with the implicit confirmations: the simulator decides to correct an error only if (i) the system prompt is not an explicit confirmation, and (ii) the simulator has already generated at least one response (this condition is necessary because the SR_{obt} is created by the analysis of the previous response of the simulator; therefore, the first time the simulator generates a response there is no SR_{obt} available). The second part handles the explicit confirmations: the user simulator decides to generate either affirmative or negative responses to answer explicit confirmation prompts. The dialogue system generates prompts to confirm the telephone number, post code and address immediately after the simulator has uttered each of these data items. Then, to generate the confirmation the simulator compares the utterance's correct semantic representation with the semantic representation obtained by the dialogue system and decides on the confirmation to be generated. At the end of the dialogue, the system

generates a prompt to confirm the products ordered. To answer this prompt, the simulator compares the product orders in the scenario against those understood by the dialogue system, accessing the dialogue history stored in the user simulator memory module, and decides on the appropriate response. The third part of the algorithm is concerned with rectification of a product order error: the simulator decides its next response, uttering again the data items missing from product orders due to deletion recognition errors. The fourth part regards the management of scenario goals: the simulator decides the new semantic representation (scenario goal) using the prompt-goal correspondence rules, which are analyzed to obtain the data items. It then prepares the appropriate responses to answer possible error correction prompts in the dialogue system. The fifth part of the algorithm concerns the response generator module: the simulator uses a particular semantic representation to search for two associated files, one containing a sentence (in text format) and the other containing the voice signal file recorded by a speaker who read the sentence. The system prompt and the sentence are stored in a log file that stores the conversation, and the voice signal file is the input for the speech recognizer of the dialogue system. The sixth part regards the simulator memory module, which stores the semantic representations obtained by the dialogue system. At the end of the algorithm, the semantic representation obtained is considered the correct semantic representation (semantic representation$_{cor}$), so that in the following interaction, the simulator can compare it against the one obtained by the dialogue system.

An example of dialogue (translated from Spanish into English) generated using the sample scenario shown in Section 2.1 is presented below.

Dialogue 4. Example of a generated dialogue:

(1) SAPLEN: Good evening, welcome to our restaurant. This is our automatic telephone service system. We inform you that the minimum price for the home delivery service is six euros. Please speak after the tone. What would you like to have?
(2) SIMULATOR: One ham sandwich
Recognizer output: One ham sandwich
(3) SAPLEN: Ok. One ham sandwich. Please say your telephone number
(4) SIMULATOR: 9 5 8 2 7 5 3 6 0
Recognizer output: 9 5 8 2 7 5 6 6 0
(5) SAPLEN: Ok. I need to confirm your telephone number. Did you say 9, 5, 8, 2, 7, 5, 6, 6, 0? Please answer yes or no
(6) SIMULATOR: No
Recognizer output: No
(7) SAPLEN: Ok. Please say your telephone number again
(8) SIMULATOR: 9 5 8 2 7 5 3 6 0
Recognizer output: 9 5 8 2 7 5 3 6 0
(9) SAPLEN: Ok. I need to confirm your telephone number again. Did you say 9, 5, 8, 2, 7, 5, 3, 6, 0? Please answer yes or no
(10) SIMULATOR: Yes
Recognizer output: Yes
(11) SAPLEN: Ok. Now, please say your post code
(12) SIMULATOR: 1 8 0 0 1
Recognizer output: 1 8 0 0 1
(13) SAPLEN: Ok. I need to confirm your post code. Did you say 1, 8, 0, 0, 1? Please answer yes or no
(14) SIMULATOR: Yes
Recognizer output: Yes
(15) SAPLEN: Ok. Now, please say your address

(16) SIMULATOR: Andalucia street number 58 first floor letter e
Recognizer output: Andalucia street number 58 first floor letter e
(17) SAPLEN: Ok. I need to confirm your address. Did you say Andalucia street, number 58, first floor letter e? Please answer yes or no
(18) SIMULATOR: Yes
Recognizer output: Yes
(19) SAPLEN: Ok. Would you like to drink something?
(20) SIMULATOR: A large beer
Recognizer output: A large beer
(21) SAPLEN: Ok. One beer. Which size would you like?
(22) SIMULATOR: Large
Recognizer output: Normal
(23) SAPLEN: Ok. One normal beer. Would you like anything else?
(24) SIMULATOR: It is wrong
Recognizer output: It is wrong
(25) SAPLEN: Ok. I'll remove the normal beer. Please repeat your last order
(26) SIMULATOR: A large beer
Recognizer output: A beer
(27) SAPLEN: Ok. One beer. Which size would you like?
(28) SIMULATOR: Large
Recognizer output: Large
(29) SAPLEN: Ok. One large beer. Would you like anything else?
(30) SIMULATOR: A cheese sandwich
Recognizer output: A cheese sandwich
(31) SAPLEN: Ok. One cheese sandwich. Would you like anything else?
(32) SIMULATOR: A chocolate milkshake
Recognizer output: A chocolate milkshake
(33) SAPLEN: Ok. One chocolate milkshake. Would you like anything else?
(34) SIMULATOR: One vegetable and turkey salad
Recognizer output: One vegetable and turkey salad
(35) SAPLEN: Ok. One vegetable and turkey salad. Anything else?
(36) SIMULATOR: No
Recognizer output: No
(37) SAPLEN: Ok. I need to confirm your orders. I think you ordered one ham sandwich, one large beer, one cheese sandwich, one chocolate milkshake and one vegetable and turkey salad. Is that ok? Please answer yes or no
(38) SIMULATOR: Yes
Recognizer output: Yes
(39) SAPLEN: The price is 2.5 euros. Is that ok? Please answer yes or no
(40) SIMULATOR: Yes
Recognizer output: Yes
(41) SAPLEN: Ok. The delivery service will take 30 min approximately to get to your address. Is that ok? Please answer yes or no
(42) SIMULATOR: Yes
Recognizer output: Yes
(43) SAPLEN: Ok. Thank you very much for using our telephone service automatic system. Good night

As can be observed, the first error occurs in recognizing the telephone number (turn 4). The dialogue system prompts for an explicit confirmation and the simulator answers a negative confirmation at turn 6. At turn 7 the dialogue system initiates a sub-dialogue to rectify the error, which is achieved after the simulator's affirmative confirmation at turn 10. Now, the dialogue system prompts for the post code and for the address, and there are no errors in the recognition of these data items. At turn 20 a substitution error provokes the second error recovery sub-dialogue, which finishes at turn 29. The simulator keeps on ordering for the products in the scenario, and at turn 37 the dialogue system generates a prompt to confirm the products ordered. The simulator checks its memory module, decides that all the orders are correct and generates the explicit affirmative confirmation (turn 38). At the following prompt, the dialogue system computes the price payable and prompts for the confirmation. For this prompt, we set up a rule to simply answer as an affirmative confirmation (a real user might disagree but for simplicity this has not been taken into account). The dialogue system now searches a database to know the delivery service time corresponding to the address uttered by the simulator, and prompts for the confirmation. As for the price confirmation, the simulator always answers an affirmative confirmation to this prompt. Finally, the dialogue system generates the farewell message and the dialogue ends.

2.6. Reusability of the user simulator corpora and modules

Setting up a user simulator for a different application would require compiling a new utterance corpus, which would be a very costly task. For example, suppose we wish to create a user simulator to interact with a dialogue system for the ATIS (Air Travel Information Service) domain. Then, new sentences (text files) representing user goals must be created (using a dialogue corpus or expert knowledge about the domain) and new utterances (voice signal files) must be recorded by several speakers who read these sentences, which may be concerned with flight reservations from one city to another, city names, dates, airline names, different questions about flights, etc. It would also be necessary to create a semantic representation for every sentence, either manually or using an automatic tool. When the utterance corpus is prepared, a new scenario corpus must be created using semantic representations associated with the sentences. These semantic representations would be the new goals to be achieved by the user simulator during the interaction with the dialogue system.

On the other hand, if a user simulator is already set up, the new domain would only require adapting the components described in this paper; such a process would be much less costly. The prompt-goal correspondence rules should be adapted to interact with the new dialogue system; for example, if the system prompts: "Where did you say you want to depart from?" the simulator must use a rule that indicates the use of an utterance whose semantic representation is the city name previously uttered by the simulator. Additionally, the procedures used to handle scenario goals must be adapted to detect the new data items contained in the scenario goals. Moreover, the procedures for handling the dialogue history need to be adapted to the new domain; for example, if the dialogue system generates a prompt to confirm all the data items regarding a flight reservation, then it would be necessary to adapt the current procedure that checks the restaurant product orders. No changes would be required for the response generator, since it only takes the semantic representation provided by the simulator's dialogue manager and searches for an utterance in the corpus associated with this semantic representation. Neither is any change in the memory needed, as this only retains the conversation history by storing all the semantic representations obtained from the dialogue system. Although preparing the corpora and adapting the simulators takes time, we believe the information the technique can provide is very valuable for enhancing a dialogue system under development, by enabling different dialogue strategies and components to be tested in a simple way. For example, we could decide to carry out new experiments using different speech recognizers, different linguistic analyzers, different semantic rules for SU, different confirmation strategies, etc., without needing new experiments with real users, who may not be readily available for testing the system so many times.

3. Experiments

We used the technique proposed in this paper to evaluate the performance of the SAPLEN system using two different recognition front-ends and two different strategies for handling user confirmations. The evaluation focuses on word accuracy (WA), SR and SU, IR and TC (Danieli and Gerbino, 1995). WA is calculated as, $WA = (w_t - (w_i + w_s + w_d))/w_t$ where w_t is the total number of words in the simulator sentences (answers to prompts generated by the dialogue system), and w_i, w_s and w_d are the number of words inserted, substituted and deleted by the recognizer, respectively. Sentence recognition is calculated as $SR = S_r/S_t$, where S_r is the number of simulator sentences recognized without any error, and S_t is the total number of simulator sentences. SU is calculated as $SU = S_u/S_t$, where S_u is the number of simulator sentences correctly analyzed semantically by the linguistic analyzer of the dialogue system, e.g., the number of sentences whose obtained semantic representation match the correct semantic representation. IR is calculated as the percentage of incorrectly recognized simulator sentences that are correctly understood by the dialogue system. TC is calculated as the percentage of generated dialogues that end successfully, which happens when the dialogue system correctly understands all the goals in the scenario used by the simulator to generate the dialogue.

3.1. Recognition front-ends

The speech recognizer of the dialogue system was created using the HTK (Hidden Markov Toolkit) (Hain et al., 1999; Woodland et al., 1999) and uses acoustic models trained with the 4767 sentences from the Albayzín acoustic database (Casacuberta et al., 1991), downsampled at 8 kHz. The vocabulary size is about 2000 words, including restaurant products, names of streets, avenues, squares, etc. Two different recognition front-ends were tested, one based on several prompt-dependent grammars and the other based on just one prompt-independent grammar. The prompt-dependent recognition front-end presents the advantage of allowing fairly good recognition results, since the recognizer uses specific grammars to analyze each utterance type. Every prompt generated by the dialogue system determines the grammar required to recognize the following utterance; for example, if the system prompts for the telephone number, the following utterance is analyzed using a grammar compiled from sentences regarding telephone numbers. Thus, if the utterance is really a telephone number then it may be correctly recognized, but if the sentence is of a different type (an address, for instance) the recognizer output will be any of the recognizable telephone numbers and the address will never be recognized. Consequently, if a user speaks to a system that uses this front-end then she/he may feel uncomfortable during the interaction, as she/he may perceive that any deviation from the system prompts provokes the system to malfunction. In order to set up this front end for the evaluation, we created 23 grammars (specifically, word bigrams) by compiling sets of sentences related to product orders, user data (telephone number, post code and address), confirmations and corrections (Albesano et al., 1997; Nasr et al., 1999; Siu and Ostendorf, 2000). Taking into account the current prompt of the dialogue system, the recognizer uses one of these bigrams to analyze the utterance.

The prompt-independent recognition front-end is based on a general grammar that is used during the whole dialogue, instead of using a particular grammar associated with each system prompt. Such a grammar is independent of the prompts and is designed to recognize any kind of sentence within the application domain. The goal of this approach is to provide users with a more comfortable and natural interaction with the dialogue system. However, in this case the grammar used by the recognizer was compiled using all the words in the system vocabulary and all the possible recognizable sentences, which may lead to an increase in recognition errors and provoke a system malfunction. In order to set up this front end for the evaluation, we used the 23 prompt-dependent grammars created for the other front-end to generate 50,000 sentences with each grammar, using several tools included in the HTK package. Taking into account the

$23 \times 50,000 = 1,150,000$ sentences generated, we compiled a word bigram that is used by the recognizer to analyze all the utterances.

3.2. Sentence understanding

The linguistic analyzer of the system uses 45 semantic rules for SU; these rely on the detection of keywords and on certain expressions (such as greetings) in the sentences. Syntactic details that do not change the semantic meaning of sentences are ignored. The analyzer uses an IR strategy that sometimes enables it to obtain the correct semantic interpretation even if some words in the recognizer output are wrongly recognized. This strategy permits recovery from recognition errors provoked by meaningless words; as such words do not change the semantic meaning of the sentence, their insertions, deletions or substitutions (by other meaningless words) are semantically recovered by discarding these words. The strategy also rectifies Spanish gender/number recognition errors, because these lexical mistakes do not change the conceptual meaning of words.

3.3. Dialogue generation cancellation

Because of recognition errors the dialogue system may take too many turns to obtain a data item; for example, if an utterance corresponding to a telephone number contains a digit that was very badly pronounced by most of the speakers who recorded the utterance corpus, it might be very difficult for this utterance to be correctly recognized. Consequently, the dialogue system might enter into an excessively long sub-dialogue to obtain the correct telephone number. The continual re-uttering of the same data item may not be acceptable for a real user, who would probably hang up if this happened. In order to take this fact into account during dialogue generation, the simulator uses a parameter called *interaction limit* (maximum number of turns) per dialogue, so that if this limit is exceeded the generation of the current dialogue is cancelled. After studying the length of some preliminary dialogues that were successfully generated, we concluded that 30 simulator turns is an appropriate value for this parameter. To understand the reason, let us examine the dialogue example shown in Section 2.5. The length of this dialogue is 43 turns (22 dialogue system turns and 21 simulator turns). There are two sub-dialogues for error recovery: one to rectify the telephone number recognition error (turns 6–10) and the other to rectify a drink order (turns 22 to 29). In both correction sub-dialogues, the user simulator employs $2 + 4 = 6$ turns. Since all the scenarios in the corpus have the same goal structure (five product orders, a telephone number, a post code and an address), a dialogue without any repair sub-dialogue requires 15 simulator turns to make the orders, utter the user data and answer the dialogue system confirmation prompts. Taking 30 as the interaction limit means that we permit $30 - 15 = 15$ correction turns per dialogue, which we consider an ample margin for the user simulator to rectify possible errors before the dialogue is cancelled.

3.4. Experimental results for the prompt-dependent recognition front-end

Using the prompt-dependent recognition front-end, we generate one hundred dialogues for every scenario, which amounts to $100 \times 18 = 1800$ dialogues. Table 3 shows the experimental results obtained.

SU is greater than WA because of the IR technique employed by the linguistic analyzer. 82.93% of sentences containing at least one recognition error are correctly understood by the dialogue system. These errors are mostly insertions of fillers (such as *uh*) and substitutions of Spanish adjectives by others that are acoustically similar and semantically equivalent (for example, the adjective *grande*—big, singular—was often replaced by the adjective *grandes*—big, plural). The linguistic analyzer implicitly rectified these types of error since they did not change the semantic representations of utterances. By analyzing the dialogues that finished without TC, we found that the strategy used by the dialogue system to explicitly confirm data

Table 3
Results for the prompt-dependent recognition front-end using explicit confirmations

	Explicit confirmation
Word accuracy	94.82
Sentence recognition	86.99
Sentence understanding	97.78
Implicit recovery	82.93
Task completion	92.77

Table 4
Description of dialogues without TC for the prompt-dependent recognition front-end using explicit confirmations

Data item wrongly confirmed	# Dialogues	Reason for non-completion of task
(a) Phone number	32	Scenario goal not achieved
(b) Post code	33	Interaction limit exceeded
(c) Address	33	Scenario goal not achieved
(d) Ordered products	32	Interaction limit exceeded
Total	130	

sometimes does not perform correctly. The problem is that recognition errors sometimes replace affirmative confirmations by negative confirmations, or vice versa. The substitution of a negative for an affirmative confirmation provokes additional dialogue turns to achieve the data items, which may not be a severe problem if the interaction limit is not exceeded. However, the substitution of an affirmative confirmation for a negative one means that the dialogue system assumes an erroneous data item to be correct, which is really a problem. This error was observed in 130 of the dialogues generated, which were cancelled or which ended with at least one incorrectly understood scenario goal. Table 4 shows the dialogues used with which TC was not achieved.

The reasons for the dialogues' not achieving TC can be explained as follows:

(a) Telephone number confirmation error. When an affirmative confirmation for the phone number is substituted for a negative confirmation, the dialogue system assumes an incorrect telephone number is confirmed. As the dialogue system strategy does not consider a posterior prompt to reconfirm this data item, the dialogue ends without TC because the telephone number in the scenario is not correctly understood by the dialogue system.

(b) Post code confirmation error. When an affirmative confirmation for the post code is substituted for a negative confirmation, the dialogue system assumes an incorrect wrong post code is confirmed. The post code is used to decide the bigram used by the dialogue system to recognize the address, namely, the name of the street, avenue, square, etc. If the dialogue system uses the wrong post code, the correct address is never obtained as the recognizer uses a bigram that does not contain the uttered name. In this case, the dialogue generation enters into a continuous loop attempting to obtain the address; this causes the dialogue to be cancelled when the interaction limit is exceeded.

(c) Address confirmation error. When an affirmative confirmation for the address is substituted for a negative confirmation, the dialogue system assumes an incorrect address is confirmed. As mentioned in (a) the dialogue system does not take into account a posterior prompt to reconfirm this data. Consequently, the dialogue ends without TC because the address in the scenario is not correctly understood by the dialogue system.

(d) Ordered products confirmation error. All the product orders are implicitly confirmed during the dialogue when the dialogue system prompts for this confirmation. However, implicit confirmation errors

Table 5
Results for the prompt-dependent recognition front-end using re-prompting confirmations

	Re-prompting confirmation
Word accuracy	96.21
Sentence recognition	88.77
Sentence understanding	99.15
Implicit recovery	92.4
Task completion	98.22

may lead the system to misunderstand the product order. For this confirmation prompt, the simulator checks the product orders in its memory module and generates a negative or an affirmative confirmation accordingly. The error found in the dialogues analyzed is that all the products orders were correct and the simulator generated an affirmative confirmation, but sometimes this confirmation was replaced by a negative confirmation. In this case, the system inquires whether the user wants to cancel the orders and start ordering again (this is not the optimal strategy for the dialogue system, but is the one initially defined in order to simplify its design). For this prompt, the simulator always generates an affirmative confirmation in order to try again to achieve the goals in the scenario. As the re-ordering procedure requires additional simulator turns, on some occasions the interaction limit is exceeded. In particular, 32 generated dialogues ended without TC because of this reason.

Taking into account problems (a)–(d), it is clear that the confirmation strategy must be improved. Therefore, we used the simulation technique to evaluate the dialogue system when it uses another confirmation strategy, called *re-prompting confirmation* in this paper. This new strategy works as follows: instead of prompting for a *yes/no* answer to confirm a data item, the dialogue system prompts again for the same data item and compares the semantic representation obtained with the previous one. If the two match, then the data item is considered confirmed; if not, the dialogue system re-prompts to obtain the data item. This procedure continues until the two previous s match. We again generated one hundred dialogues for every scenario using this new confirmation strategy. Table 5 shows the experimental results obtained.

Comparison of Tables 3 and 5 shows that the scores for this new confirmation strategy are higher than those of the previous one, which is because this is a *stronger* way to confirm data items, being based on the comparison of semantic representations built from more than just one keyword in utterances; thus, if one keyword is wrong then the semantic representation built from it is also wrong. For example, the semantic representation regarding a post code is built from an utterance that contains five keywords (digits), for instance, "1 8 0 1 2"; if a recognition error then substitutes, deletes or inserts a digit, the semantic representation created is changed. Consequently, to have a post code wrongly confirmed requires the same errors to occur in the recognition of two consecutive utterances. Of course, this is not impossible but it is much less likely to occur than just a substitution of a keyword ("yes" for "no", or vice versa), which is the problem with the explicit confirmation strategy.

3.5. Experimental results for the prompt-independent recognition front-end

Finally, we evaluated the dialogue system when it used the prompt-independent recognition front-end, considering both confirmation strategies. Table 6 describes the results obtained.

In this case, all the scores are lower since this front-end always uses the same word bigram to analyze any utterance, which covers all the words in the system vocabulary and all the recognizable sentences. Consequently, in this case the recognizer outputs often contain insertions of words that have nothing to do with the current system prompt, for example, the system prompts for the telephone number and the recognizer

Table 6
Results for the prompt-independent recognition front-end

	Explicit confirmation	Re-prompting confirmation
Word accuracy	77.98	82.25
Sentence recognition	75.11	80.49
Sentence understanding	78.7	83.6
Implicit recovery	26.75	32.5
Task completion	72.11	84.24

output is "nine two beer one ham five street zero four". Focusing on the explicit confirmation strategy, and comparing Tables 3 and 6, it can be observed that TC increased by 20.66 points when the prompt-dependent recognition front-end was used. Focusing on the re-prompting confirmation strategy, and comparing Tables 5 and 6, we see that TC increased by 13.98 points when the prompt-dependent recognition front-end was used. These results show that this front-end is in theory preferable but that this is only so if the user answers the system prompts exactly as the dialogue system expects. The real case is that users, especially inexperienced ones, answer the same prompt differently; for example, they may say simply "yes" or "I said 9, 5, 8, 1, 2, 4, 3, 2, 1, 2" to confirm a telephone number. Although the first response is the expected one and is likely to be correctly understood by the dialogue system, the second one will cause the system to malfunction. Hence, in order to increase the robustness of the system we should think about enhancing the performance when it uses the prompt-independent recognition front-end. Table 6 also shows higher scores for the re-prompting strategy. In this case, however, the problem regarding explicit confirmations is not only concerned with *yes–no* substitutions, which is the problem for the other front-end, but also with word insertions. These errors confuse the dialogue system, as the linguistic analyzer cannot implicitly rectify confirmation utterances containing keywords other than "yes" or "no". Consequently, the differences in the experimental results for the two confirmation strategies were greater when the system used the prompt-independent recognition front-end.

4. Conclusions and future work

In real conversations, users may behave unexpectedly when they interact with a dialogue system; for example, they may hesitate, get confused or answer the system prompts with information the system is not really prompting for. The simulator presented in this paper, however, always provides the information prompted for by the dialogue system, has no doubts during interaction and never gets confused or nervous because an automatic system is involved. Consequently, the experimental results presented in this paper may not be very realistic, as they were obtained considering dialogues generated by a simplified model of system–human interaction. The results would doubtless be worse for a real system-user interaction in which the user is not an expert who knows how to communicate with the system. Moreover, the present results were obtained using utterances recorded under quiet lab conditions; in a real interaction, noise and other phenomena (cross-talk, silence, etc.) would decrease the performance of the dialogue system.

Despite the above, the technique proposed in this paper was useful for evaluating a dialogue system using two different recognition front-ends and two different dialogue strategies to handle user confirmations. On the one hand, the evaluations prove that the prompt-dependent recognition front-end achieves better results than the prompt-independent one. The latter is more appropriate, nevertheless, if we wish to provide users with a more natural and comfortable interaction because it allows them to utter any recognizable utterance at any moment, irrespective of what the dialogue system is prompting for. On the other hand, the evaluations show that the explicit confirmation strategy sometimes fails and degrades the

performance of the dialogue system, while the re-prompting confirmation strategy enhances the performance of both recognition front-ends. These conclusions suggest we should use the prompt-independent recognition front-end (to provide a natural interaction) and the re-prompting confirmation strategy (to reduce confirmation errors); doing this, however, would require enhancing the recognition front-end. One way to improve it would be by including context information in the recognition phase so that the probabilities of the most likely sentences for a given system prompt are increased. Then, for example, if the system prompted for the telephone number, the user would be free to utter any recognizable sentence that, in theory, would be recognized correctly. However, in this case the context information would indicate that the transition probabilities between digits should be increased so that a telephone number would be more likely to be the recognizer output than a food order.

There remain several issues that should be addressed to improve the technique presented in this paper, such as how to include new types of goals in the scenarios (when the user changes his/her mind, etc.). Moreover, it should be possible to refine the technique by including more data in the scenarios and obtaining better models of real users. For example, probabilities would be associated with the goals to vary the way the user simulator answers the prompts, in an attempt to model the fact that users sometimes get confused and answer the prompts with the wrong information. The probabilities could also allow the simulator to generate typical errors in spoken interactions (for example, overlapping, silence or non linguistic input). Another way to enhance the technique would be by using an interaction limit not just for the whole dialogue but also for the sub-dialogues, so that if the system prompts repeatedly for the same data item, the simulator cancels the dialogue. This behavior would model the fact that a real user may hang-up if she/he considers the performance of the dialogue system unacceptable. One more way to refine the technique would be by recording and using utterances representing users' different feelings about the interaction with the dialogue system, for example whether it is pleasant or annoying, represented by different intonations. The simulator could start by using a "normal" intonation to interact with the dialogue system, and then change to an "annoyed" intonation if the dialogue system makes too many errors. This would make it possible to model the fact that users might change their linguistic behavior during the interaction. When we modify the system to test these new strategies we can employ the simulation technique as a simple test of the effects on the performance of the dialogue system, saving the time and costs otherwise required to search for real users to test the system.

References

Albesano, D., Baggia, P., Danieli, M., Gemello, R., Gerbino, E., Rullent, C., 1997. DIALOGOS: a robust system for human–machine spoken dialogue on the telephone. In: Icassp'97, Munich, Germany, pp. 1147–1150.

Allen, J., 1995. Natural language understanding. The Benjamin/Cummings Publishing Company Inc.

Araki, M., Watanabe, T., Doshita, S., 1997. Evaluating dialogue strategies for recovering from misunderstandings. In: Proc. IJCAI Workshop on Collaboration Cooperation and Conflict in Dialogue Systems, pp. 13–18.

Araki, M., Doshita, S., 1997. Automatic evaluation environment for spoken dialogue system. In: Mayer, S., Mast, E., LuperFoy, M. (Eds.), Dialogue Processing in Spoken Language Systems. Springer, pp. 183–194.

Boros, M., Heisterkamp, P., 1999. Linguistic phrase spotting in a single application spoken dialogue system. In: Eurospeech'99, Budapest, Hungary, pp. 1983–1986.

Bouwman, G., Hulstijn, J., 1998. Dialogue strategies redesign with reliability measures. In: First Int. Conf. on Language Resources and Evaluation, Granada, Spain, pp. 191–198.

Buntschuh, B. et al., 1998. VPQ: a spoken language interface to large scale directory information. In: Eurospeech'97, Rhodes, Greece, pp. 815–818.

Casacuberta, F., García, R., Llisterri, J., Nadeu, C., Pardo, J.M., Rubio, A., 1991. Development of Spanish corpora for speech research (ALBAYZIN). In: Workshop on International Cooperation and Standardization of Speech Databases and Speech I/O Assesment Methods, Chiavari, Italy, 26–28 September.

Danieli, M., Gerbino, E., 1995. Metrics for evaluating dialogue strategies in a spoken language system. In: AAAI Spring Symp. on Empirical Methods in Discourse Interpretation and Generation, pp. 34–39.

Gorin, A.L., Riccardi, G., Wright, J.H., 1997. How may I help you? Speech Commun. 23, 113–127.
Hain, T., Woodland, P.C., Niesler, T.R., Whittaker, E.W.D., 1999. The 1998 HTK system for transcription of conversational telephone speech. In: ICASSP'99, Phoenix.
Kellner, A., Rueber, B., Seide, F., Tran, B.H., 1997. PADIS—an automatic telephone switchboard and directory information system. Speech Commun. 23, 95–111.
Kellner, A., Rueber, B., Schramm, H., 1998. Strategies for name recognition in automatic directory assistance systems. In: IVTTA'98, Torino, Italy, pp. 21–26.
Lavelle, C.A., De Calmès, M., Pérennou, G., 1999. Confirmations strategies to improve correction rates in telephonic inquiry dialogue system. In: Eurospeech'99, pp. 1399–1402.
Lamel, L.F. et al., 1997. The LIMSI RailTel system: field trial of a telephone service for rail travel information. Speech Commun. 23, 67–82.
Lamel, L.F. et al., 1998. User evaluation of the Mask kiosk. In: ICSLP'98, Sydney, Australia, pp. 2875–2878.
Levin, E., Pieraccini, R., Eckert, W., 2000. A stochastic model of human–machine interaction for learning dialog strategies. IEEE Trans. Speech Audio Process. 8, 11–23.
López-Cózar, R., Rubio, A.J., García, P., Segura, J.C., 1998. A spoken dialogue system based on a dialogue corpus analysis. In: LREC 1998, Granada, Spain, pp. 55–58.
López-Cózar, R., Rubio, A.J., García, P., Segura, J.C., 1999. A new word-confidence threshold technique to enhance the performance of spoken dialogue systems. In: Eurospeech'99, Budapest, Hungary, pp. 1395–1398.
López-Cózar, R., Rubio, A.J., Díaz Verdejo, J.E., De la Torre, A., 2000a. Evaluation of a dialogue system based on a generic model that combines robust speech understanding and mixed-initiative control. In: LREC 2000, Athens, Greece, pp. 743–748.
López-Cózar, R., Rubio, A.J., Benítez, M.C., Milone, D.H., 2000b. Restricciones de Funcionamiento en Tiempo Real de un Sistema Automático de Diálogo (Real-Time Performance of a Spoken Dialogue System). SEPLN J. (Spanish Society for Natural Language Processing) 26, 169–174.
Nasr, A., Esteve, Y., Béchet, F., Spriet, T., de Mori, R., 1999. A language model combining *N*-grams and stochastic finite state automata. In: Eurospeech'99, Budapest, Hungary, pp. 2175–2178.
Niimi, Y., Nishimoto, T., 1999. Mathematical analysis of dialogue control strategies. In: Eurospeech'99, Budapest, Hungary, pp. 1403–1406.
Noeth, E., Boros, M., Haas, J., Warnke, V., Gallwitz, F., 1999. A hybrid approach to spoken dialogue understanding: intonation, statistics and partial parsing. In: Eurospeech'99, Budapest, Hungary, pp. 2019–2022.
Rabiner, L., Juang, B.H., 1993. Fundamentals of Speech Recognition. Prentice-Hall.
Rosset, S., Bennacef, S., Lamel, L., 1999. Design strategies for spoken language dialog systems. In: Eurospeech'99, Budaest, Hungary, pp. 1535–1538.
Schadle, I., Antoine, J.Y., Memmi, D., 1999. Connectionist language models for speech understanding: the problem of word order variation. In: Eurospeech'99, Budapest, Hungary, pp. 2035–2038.
Siu, M., Ostendorf, M., 2000. Variable *N*-grams and extensions for conversational speech language modeling. IEEE Trans. Speech Audio Process. 8, 63–75.
Souvignier, B., Kellner, A., Rueber, B., Schramm, H., Seide, F., 2000. The thoughtful elephant: strategies for spoken dialog systems. IEEE Trans. Speech Audio Process. 8, 51–62.
Swerts, M., Ostendorf, M., 1997. Prosodic and lexical indications of discourse structure in human–human interactions. Speech Commun. 22, 25–41.
Woodland, P.C., Hain, T., Moore, G.L., Niesler, T.R., Provey, D., Tuerk, A., Whittaker, E.W.D., 1999. The 1998 HTK broadcast news transcription system: development and results. DARPA Broadcast News Workshop, March 1999 (Herndon, VA).
Zue, V. et al., 1997. From Interface to content: translingual access and delivery of on-line information. In: EuroSpeech'97, Rhodes, Greece, pp. 2227–2230.
Zue, V. et al., 2000. JUPITER: a telephone-based conversational interface for weather information. IEEE Trans. Speech Audio Process. 8, 85–95.