Artif Intell Rev (2006) 26:291–323
DOI 10.1007/s10462-007-9059-9

Testing the performance of spoken dialogue systems by means of an artificially simulated user

Ramón López-Cózar · Zoraida Callejas · Michael McTear

Published online: 23 November 2007
© Springer Science+Business Media B.V. 2007

Abstract This paper proposes a new technique to test the performance of spoken dialogue systems by artificially simulating the behaviour of three types of user (very cooperative, cooperative and not very cooperative) interacting with a system by means of spoken dialogues. Experiments using the technique were carried out to test the performance of a previously developed dialogue system designed for the fast-food domain and working with two kinds of language model for automatic speech recognition: one based on 17 prompt-dependent language models, and the other based on one prompt-independent language model. The use of the simulated user enables the identification of problems relating to the speech recognition, spoken language understanding, and dialogue management components of the system. In particular, in these experiments problems were encountered with the recognition and understanding of postal codes and addresses and with the lengthy sequences of repetitive confirmation turns required to correct these errors. By employing a simulated user in a range of different experimental conditions sufficient data can be generated to support a systematic analysis of potential problems and to enable fine-grained tuning of the system.

Keywords Spoken dialogue systems · Speech recognition · Speech understanding · User simulation · Artificial intelligence · Natural language processing · Robust human–computer interaction

---

R. López-Cózar (✉) · Z. Callejas
Department of Languages and Computer Systems, Computer Science Faculty, Granada University, Granada 18071, Spain
e-mail: rlopezc@ugr.es

Z. Callejas
e-mail: zoraida@ugr.es

M. McTear
School of Computing and Mathematics, University of Ulster, Shore Road, Newtownabbey, Northern Ireland, UK
e-mail: mf.mctear@ulster.ac.uk

1 Introduction

A spoken dialogue system enables a human user to communicate with a computer to obtain information or engage in some form of transaction, which can range from a business service such as banking or making a travel reservation to more informal interactions such as casual conversation or playing an interactive game. Spoken dialogue technology has been an active research area for the past two decades. A number of academic and commercial systems have been developed, which demonstrate the potential of this technology (McTear 2004; Möller 2004, López-Cózar and Araki 2005).

The design of an effective dialogue management component is a central aspect of dialogue system engineering (Pieraccini and Huerta 2005). Essentially the aim is to create a system that successfully addresses a number of disparate performance criteria, including user satisfaction, task success, and minimization of dialogue cost factors such as transaction times and error correction rates (Walker et al. 1997). However, this is by no means a simple task, as there are many factors that can affect the performance of a system. These can be classified roughly into the following three types: system performance factors, user factors, and dialogue strategy factors.

One of the most obvious system performance factors is the speech recognition error rate. Although a high rate of recognition error does not necessarily entail a low transaction success rate, in general interactions in which there are many recognition failures, particularly in respect of significant keywords, are likely to lead to unsatisfactory dialogues because of the need for the user to repeat or rephrase and for the system to engage in correction subdialogues. Given that recognition is unlikely to be perfect in current systems, there is a need for robust error correction strategies. It is usually a task for the designer of a system to choose particular error correction strategies based on experience and on iterative field testing of a system.

User factors include the different ways that a user might interact with a system, determined on the one hand by user preferences (for example, for a system directed as opposed to a mixed initiative dialogue), and on the other hand by factors arising out of the user's particular interactive style—for example, whether the user is co-operative or not, is willing to accept certain system limitations, requires information quickly, and so on.

Finally, there are a number of different ways of implementing dialogue strategies. In addition to the choice between system initiative and mixed initiative, there are choices between different methods for verification of the user input, ranging from explicit confirmation through implicit confirmation to no confirmation at all (Litman and Pan 2002). In addition to this, system prompts can be highly constraining or can be open-ended, with implications for the types of recognition grammars required—on the one hand, restricted grammars, on the other more open-ended to cover a wide range of potential input (Singh et al. 2002). As with the other factors mentioned, decisions about which strategies to implement are usually a matter of the designer's choice.

Given this wide variety of factors that can influence how a system is designed and how it will perform, there has emerged a recent trend involving the application of machine learning algorithms in order to learn optimal dialogue management policies. The expected benefits of this approach in comparison with hand-crafted and knowledge-based methods are a reduction in development time, greater robustness, and less developer bias. More specifically, the dialogue system is modelled as a Markov Decision Process (MDP) in which all the states of the system as well as the choices at each state giving rise to state transitions are modelled, and reinforcement learning is applied to find the optimal dialogue policy (Walker 2000; Levin et al. 2000; Young 2002).

However, whether the factors that determine a system's performance are anticipated by the designer and addressed through hand-crafted methods or whether they are explored using reinforcement learning, the problem remains of how to collect sufficient data to support either of these approaches. This problem is closely related to the type of dialogue management used by the system, in as much as simpler management methods will generally result in fewer performance factors to consider.

The simplest type of dialogue manager can be specified as a finite state machine in which all the dialogue call-flows are pre-determined and represented in the form of a graph. This type of dialogue control usually implies a system-directed dialogue in which the system retains initiative in the dialogue and the user is constrained to responding minimally to carefully designed system prompts (McTear 2004). A slightly more complex control strategy allows the user to respond with more than one item of information at a time—for example, in VoiceXML mixed-initiative applications where the information elicited from the user is recorded in a form and the Form Interpretation Algorithm (FIA) allows these data to be input in any order and in different combinations (Pieraccini and Huerta 2005). Finally more advanced architectures permit relatively unrestricted input from the user and the control of the dialogue is determined opportunistically on a turn-by-turn basis rather than as a result of a dialogue script (Allen et al. 2001).

Even with simple dialogue management techniques such as finite-state control the range of potential call-flows can quickly grow to unmanageable proportions due to the large number of possible system states and the even larger number of possible state transitions. To take a simple example: for a system that is required to elicit 5 items of information from the user, where each item can have one of the values {unknown, to_ be_ confirmed, confirmed), the number of dialogue states would be 3^5, i.e. 243. Indeed, as Aust and Oerder (1995) pointed out with reference to the Philips train timetable information system, there were about 1,000 system questions to which the user could respond in a variety of different ways, each requiring different system responses, so that a graph representing all possible call-flows would require tens of thousands of transitions.

In the case of advanced dialogue systems, the possible paths through the dialogue state space are not known in advance and specification of all possible transitions is not possible. In reinforcement learning this issue has been addressed by constraining the state space to a manageable size and by focusing on task-oriented systems in which the goal is to elicit a finite (generally fairly small) set of values from the user to fill the slots in a form. Currently there do not appear to be methods for exhaustively searching the complete state space of a more advanced dialogue system in which the state space is emergent rather than pre-determined.

One possible way to address some of these issues is to collect and analyze vast amounts of data covering different ways that users interact with a system and the different choices that can be applied in dialogue management. However, controlling all these factors with real users in actual interactions would be a daunting, if not impossible task. A more efficient method for collecting data under controlled conditions would be to simulate interactions in which the various user and system factors can be systematically manipulated.

In this paper a system for testing the performance of spoken dialogue systems using a simulated user is presented and evaluated under a number of different conditions. The remainder of the paper is organised as follows. Section 2 presents a survey of previous studies related to simulated users that includes the purposes, implementation approaches and evaluation of dialogue simulations. Section 3 presents a brief description of the Saplen system used in the experiments, addressing the dialogue corpus employed to design it and the kind of dialogue management implemented. Section 4 discusses briefly the simulated user that we developed in a previous study, which is the basis for the technique proposed in this paper.

Section 5 describes the proposed technique addressing differences and similarities with other simulated users, and focusing on the simulation of different types of user. Next the section focuses on the implementation of the technique, addressing the generation of responses for explicit and implicit system confirmations, the management of scenario goals, the generation of responses to provide missing data, the selection of utterances from the corpus and the preparation for the next interaction. The section concludes with a sample dialogue between the Saplen system and the simulated user behaving as a cooperative user. Section 6 focuses on the experimental results. It first discusses the evaluation measures and then describes the utterances corpus, the two kinds of language model employed for speech recognition, and the set of scenarios used for the automatic generation of dialogues. The section concludes by presenting and discussing the results obtained. Section 7 discusses limitations of the proposed technique and finally Section 8 presents some conclusions and some proposals for future work.

2 Previous related work

The design of dialogue systems is a complex task that generally requires the use of expert knowledge acquired in the development of previous systems, including tests taken with users interacting with the system. The development of these systems mainly involves an iterative process in which different prototypes are released and tested with real users. From these tests, objective and subjective measures can be obtained concerning the appropriateness of several aspects of the system (Möller 2004). The tests provide a basis for refinements of the prototype systems until eventually a system is obtained which is as perfect as possible in terms of correct functioning and user satisfaction. However, using user studies to support the development process is very expensive and time consuming. The employment of techniques like Wizard of Oz (Dow et al. 2005; Carbini et al. 2006) for system design reduces the costs by avoiding the need for a functional prototype for testing purposes, but unfortunately it does not avoid the need for constant human intervention to obtain useful results. For these reasons, during the last decade many research groups have been attempting to find a way to automate these processes, leading to the appearance of the first simulated users in the late 90s. These are automatic systems able to simulate user behaviour when interacting with a dialogue system.

2.1 Purposes of simulated users

The main purpose of a simulated user is to improve the usability of a spoken dialogue system through the generation of corpora of interactions between the system and simulated users (Möller et al. 2006). Collecting large samples of interactions with real users is an expensive process in terms of time and effort. Moreover, each time changes are made to the system it is necessary to collect more data in order to evaluate the changes. Thus the availability of large corpora of simulated data should contribute positively to the development of dialogue systems (Chung 2004).

Simulated data can be used to evaluate different aspects of a dialogue system, particularly at the earlier stages of development, or to determine the effects of changes to the system's functionalities. For example, in order to evaluate the consequences of the choice of a particular confirmation strategy on transaction duration or user satisfaction, simulations can be done using different strategies and the resulting data can be analysed and comparisons made between the strategies employed. Another example would be the introduction of errors or unpredicted answers in order to evaluate the capacity of the dialogue manager to react to unexpected situations.

A second usage is to support the automatic learning of optimal dialogue strategies using reinforcement learning. Large amounts of data are required for a systematic exploration of the dialogue state space and corpora of simulated data are extremely valuable for this purpose, given the costs of collecting data from real users. In any case, it is possible that the optimal strategy may not be present in a corpus of dialogues gathered from real users, so additional simulated data may enable additional alternative choices in the state space to be explored (Schatzmann et al. 2005a).

The most common usage for a simulated user is to take the place of real users and to engage in an interaction with a dialogue system by generating responses to system prompts. The user utterances may be generated from an existing corpus of dialogue data in conjunction with a set of dialogue scenarios requiring particular types of input from the user in response to system prompts (see, for example, López-Cózar et al. 2003). In some cases the simulated input will model user errors. At the simplest level these may include failure to respond as well as errors of vocabulary and grammar. More complex user errors have been modelled by Möller et al. (2006). These include: state errors, where the user provides input that is invalid in the current dialogue state but would be valid in some other dialogue state; capability errors, where the user issues a command or asks for information that is not supported by the system; and modelling errors, where the input is invalid in respect of how the world is represented in the system. These error types are incorporated along with correct user inputs in an executable user interaction model that generates user input using a probabilistic finite state automaton.

Another use of simulation is to model both the user and the system. Cuayáhuitl et al. (2005) describe a simulation method in which a training set of real dialogues is used to acquire knowledge and train the system and user models. The system model is a probabilistic dialogue model manager that controls the dialogue flow while the user model is a probabilistic model that simulates user responses. Both models interact in order to generate simulated dialogues and the dialogues that are generated are compared to a set of real dialogues in order to determine the realism of the simulated dialogues.

2.2 Approaches to the implementation of simulated users

We can distinguish two main approaches to the creation of simulated users: rule based and data or corpus based. In a rule-based simulated user the investigator can create different rules that determine the behaviour of the system (Chung 2004; Lin and Lee 2001; López-Cózar et al. 2003). This approach is particularly useful when the purpose of the investigation is to evaluate the effects of different dialogue management strategies. For example: the system can be set up to react to simulated user errors in a number of different ways and comparisons between the effects of these different strategies can be made. In this way the investigator has complete control over the design of the evaluation study.

An alternative approach—often described as corpus-based or data-based—uses probabilistic methods to generate the user input, with the advantage that this uncertainty can better reflect the unexpected behaviours of users interacting with the system (Eckert et al. 1997; Scheffler and Young 2000; Georgila et al. 2005; Möller et al. 2006). Earlier studies have involved the use of the ‘bigram’ model of dialogue in which the simulated user input is dependent only on the previous system utterance (Eckert et al. 1997). The advantage of this approach lies in its simplicity and in that it is totally domain independent. The main disadvantage, however, is that it may be too limited to give a realistic simulated behaviour because, although user actions are dependent on the previous system action, they should also be consistent throughout the dialogue as a whole.

The simulated user input can be represented in a number of ways. One of the most common ways is in the form of user intentions (or dialogue acts). This type of input is useful especially where the focus of the investigation is on the optimisation of dialogue strategies and also where issues of data sparseness would make the use of strings of words or segments of speech inapplicable, particularly for applications involving learning (Eckert et al. 1997; Levin et al. 2000; Scheffler and Young 2000, 2001). Alternatively strings of words or spoken input can be used, especially if the focus of the investigation is on the performance of the system's language understanding and recognition components. Chung (2004) makes use of simulated input at word and speech levels in order to train the understanding and recognition components, whereas these levels of input are used by López-Cózar et al. (2003) to evaluate the performance of the speech recogniser and the use of different language models, and also to investigate different confirmation strategies. Simulated speech input can be generated by passing word strings through the TTS component or, as in the López-Cózar et al. (2003) study, by retrieving recorded samples of user utterances from a corpus of dialogues involving real users interacting with the system.

Scheffler and Young (2000, 2001) proposed a goal directed model in which a goal is defined as a specification of the dialogue transaction that the user wants to accomplish and implemented as a structure of attribute-value pairs where each attribute has a status tag representing the state of each goal from the user's point of view (pending, specified, urgent or not applicable). An advantage of this approach is that it models several error types, where an error is viewed in terms of the difference between the original user intention and the interpretation captured by the system.

Cuayáhuitl et al. (2005) combine the goal oriented approach with the bigram model using Hidden Markov Models (HMMs) to predict not only the user intentions but also the system ones. Additionally, Pietquin and Dutoit (2006a, b) modify the Scheffler model by relating the user's goal to the user's behaviour during the dialogue. Finally, Georgila et al. (2005) have extended these techniques using n-grams models instead of bigrams in order to represent the complete dialogue history.

2.3 Evaluation of simulated users

The evaluation of simulated users is a complex issue (Schatzmann et al. 2005a). Evaluation has usually addressed one or more of the following issues: ease of use, whether the dialogues produced are valid in comparison with dialogues produced by real users, and whether the generated corpus is useful (Möller et al. 2006). Cuayáhuitl et al. (2005) employ three metrics to compare simulated and real dialogues: dialogue length (average number of turns), dialogue similarity (normalized distance of HMMs between two dialogue sets) and precision-recall (how well the model predicts the test and training data). In several systems evaluations have attempted to reflect user satisfaction by employing the two measures: dialogue duration and task completion (Scheffler and Young 2001; Schatzmann et al. 2005b; Pietquin and Dutoit 2006b). Some authors have employed more fine-grained measures relating the quality of the obtained dialogue—for example, how much information is transmitted in each turn or how active or cooperative the users are (Schatzmann et al. 2005b). Moreover, as argued by Schatzmann et al. (2005b), in addition to measuring validity in terms of whether the model generates outputs similar to a human's response, the simulation should also aim to reproduce sufficient variety in human behaviour to represent the whole user population and not only an average user.

While the simulated users described here are concerned with spoken dialogue systems, there is an increasing number of systems being developed that include additional modalities in combination with speech. In these cases more complex simulations are likely to be required to model the actions of users as well as their utterances (Okamoto et al. 2005).

3 The Saplen dialogue system

This section briefly describes the Saplen dialogue system used in the experiments. Saplen was originally implemented to answer Spanish telephone-based orders and queries by clients of fast food restaurants (López-Cózar et al. 1997). Figure 1 shows the relationship between this system and the simulated user used as the basis for this study, which is described in more detail in Sect. 4. The speech synthesiser of the system is not shown as it is not used in the experiments presented in this paper.

The system employs an HTK-based speech recogniser (Hain et al. 1999; Young et al. 2000) that generates as output an N-best list of recognition hypotheses (N = 10). A confidence score $C(w)$ in the range 0.0–1.0 is assigned to each word w in the hypothesis according to the following expression:

$$C(w) = \frac{\sum_{i \in I_w} \exp(-sc_i)}{\sum_{j=1}^{N} \exp(-sc_j)}$$

where sc_j represents the score (log probability) of the jth hypothesis in the N-best list, I_w represents the set of indices of hypotheses that contain the word w ($I_w \subset \{1, \ldots, N\}$), and sc_i represents the score of the ith hypothesis that contains w. For example: for the food order “I would like to have three green salads” the recognition hypothesis may be “I (0.7590) would (0.6982) like (0.9268) to (0.4285) have (0.6929) six (0.3974) green (0.7059) salads (0.8182)”, which is incorrect as the word “three” is substituted by the word “six”. The system employs two confidence thresholds ($T_1 = 0.3$ and $T_2 = 0.5$) to decide whether to discard or confirm either explicitly or implicitly each word w in the recognition hypothesis. The word w is discarded if $C(w) < T_1$, is confirmed explicitly if $T_1 \leq C(w) < T_2$ and is confirmed implicitly if $T_2 \leq C(w)$.

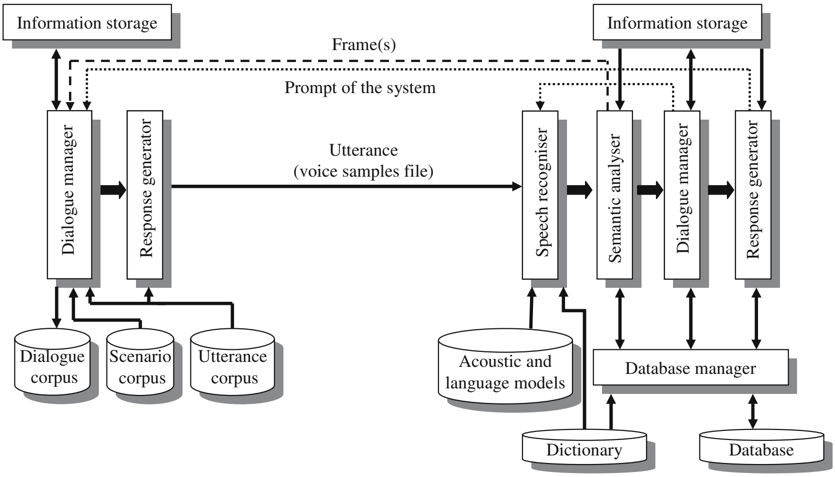

Fig. 1 Relationship between the simulated user and the Saplen system

3.1 Dialogue corpus

To develop the system we compiled a dialogue corpus in a fast food restaurant that contains about 800 recorded dialogues in Spanish involving conversations between clients and restaurant assistants (López-Cózar et al. 1998). These dialogues contain product orders, telephone numbers, postal codes, addresses, queries, confirmations, greetings and other types of utterances. The dialogues were transcribed, labelled and analysed to include tags regarding the speakers (clients and restaurant assistants), utterance types, semantic information about the utterances, and other kinds of information. The corpus contains around 5,500 utterances and about 2,000 words which were used for previous studies (López-Cózar et al. 2000, 2002, 2003; López-Cózar and Milone 2001; López-Cózar and Callejas 2005). The semantic information associated with each utterance is represented as a frame (Allen 1995) as explained in Section 4.1.

3.2 Dialogue management

The dialogue management in the Saplen system is as follows: after an initial welcome message the system prompts the user to enter product orders or queries, explaining that orders and queries must be entered one by one. When a product order has been made the system prompts for the telephone number of the user and confirms the recognised number. The system checks whether the number is in a database of known clients. If it is, it then prompts the user to confirm the stored data, otherwise it prompts for the user's postal code and confirms it. The system then prompts for the user's address and confirms the data obtained. If the user confirms them, they are stored in the users database together with the telephone number. If the data are not confirmed the system tries to get the correct data by asking for just one item of address data at a time, i.e. address type, address name, building number, building floor and apartment letter.[1] At the end of the dialogue the system confirms the products ordered, states the total price to be paid and prompts the user to accept it. It also gives the estimated delivery time and prompts the user to accept it. The dialogue ends with the system thanking the user for the call. All the confirmation prompts generated by the system are "yes/no" questions, as for example: "Let's see. I think you ordered a ham sandwich, a Spanish omelette sandwich, two salads and two large beers. Is this correct? Please say yes or no".

4 Previous version of our simulated user

This section briefly describes the simulated user that we developed in a previous study (López-Cózar et al. 2003) which is the basis for the work presented in this paper. The purpose of this simulated user is to interact automatically with the Saplen system in a spoken conversation, and to obtain a dialogue corpus suitable for testing the performance of the system. As can be observed in Fig. 1, the simulated user receives the current prompt generated by the system as well as the frame(s) obtained by the system from the analysis of its previous response. This response is an utterance (voice samples file) recorded by a client of the fast food restaurant, which is taken from the dialogue corpus discussed in Sect. 3.1.

---

[1] These are the data typically used in Spain for the postal address, such as "Avenida de Andalucía, número 24, portal 4, tercero E", but there are also addresses that only include address type (e.g. "street"), address name (e.g. "Elm") and building number (e.g. "13"). In the scenarios used to test the proposed technique, described in Sect. 6.4, all the addresses considered include all the data items, to obtain experimental results in the hardest case.

# the user wants to order a ham sandwich

(<AMOUNT> = ”1”,
<FOOD> = ”SANDWICH”,
<INGREDIENTS> = ”HAM”)

# the user wants to order a cheese sandwich

(<AMOUNT> = ”1”,
<FOOD> = ”SANDWICH”,
<INGREDIENTS> = ”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”,
<INGREDIENTS> = ”VEGETABLE TURKEY”)

# the user telephone number

(<TELEPHONE_NUMBER> = ”958275360”)

# the postal code of the user

(<POSTAL_CODE> = ”18001”)

# the user address

(<ADDR_TYPE> = ”STREET”,
<ADDR_NAME> = ”ANDALUCIA”,
<ADDR_NUMBER> = ”58”,
<ADDR_FLOOR> = ”FIRST”,
<ADDR_LETTER> = ”E”)

Fig. 2 A sample scenario for the simulated user

Therefore, the frame(s) can be affected by speech recognition errors just as if the analysed response had been uttered by a real user.

To interact with the system the simulated user employs a set of scenarios that indicate the goals it must try to achieve during the dialogue. For example, the scenario shown in Fig. 2 indicates that the simulated user must order “one ham sandwich”, “one cheese sandwich”, “one large beer”, “one chocolate milkshake” and “one vegetable and turkey salad”. Hence, when the system prompts the simulated user to make product orders and enter the telephone number, postal code and address of the user, the simulated user must use (i.e. *utter*) the product orders, telephone number, postal code and address in the scenario. As will be explained in Sect. 6.4, these scenarios are created by selecting utterances in a test utterance corpus.

To answer a system prompt, after the simulated user has selected the appropriate scenario goal, e.g. (<POSTAL_CODE> = “18001”), it selects at random an utterance in the corpus, e.g. “uhm my postal code is 18001”, for which the associated reference frame matches the selected goal, as will be explained in Sect. 5.3. This utterance is the input for the speech recogniser of the system. An example of an interaction between the simulated user and the Saplen system (translated from Spanish into English) is shown in Dialogue 1. In this dialogue the simulated user behaved as a very cooperative user (discussed in Sect. 5.2) and the system employed the prompt-dependent language models (PDLMs) to be discussed in Sect. 6.3.

(1) *Saplen*: Good evening, welcome to our restaurant. This is our automatic telephone service system. The minimum price for the home delivery service is six euros. Please speak after the tone. What would you like to have?
(2) *Simulated user*: Three ham sandwiches [Recognition hypothesis: six (0.3285) ham (0.5956) sandwiches (0.8510)]
(3) *Saplen*: I have to confirm your order. Did you say six ham sandwiches? Please say yes or no.
(4) *Simulated user*: No [Recognition hypothesis: no (0.6734)]
(5) *Saplen*: OK. How many sandwiches did you say?

(6) *Simulated user*: Three [Recognition hypothesis: three (0.7624)]
(7) *Saplen*: OK. So that's three ham sandwiches. Please say your telephone number.
(8) *Simulated user*: 9 5 8 2 7 5 3 6 0 [Recognition hypothesis: nine (0.6851) five (0.6985) eight (0.7659) two (0.7565) seven (0.8967) five (0.8642) three (0.4986) six (0.7521) zero (0.8634)]
(9) *Saplen*: I need to confirm your telephone number. Did you say 9, 5, 8, 2, 7, 5, 3, 6, 0? Please answer yes or no.
(10) *Simulated user*: Yes [Recognition hypothesis: yes (0.5834)]
(11) *Saplen*: OK. Telephone number 9, 5, 8, 2, 7, 5, 3, 6, 0. Please say your postal code.
(12) *Simulated user*: 1 8 0 0 1 [Recognition hypothesis: 1 (0.8561) 8 (0.7637) 0 (0.5367) 0 (0.5756) 1 (0.8985)]
(13) *Saplen*: OK. Postal code 1, 8, 0, 0, 1. Please say your address, for example Elm street, number 13, second floor, apartment letter e.
(14) *Simulated user*: Andalucia street number 58 first floor letter e [Recognition hypothesis: andalucia (0.8659) street (0.76395) number (0.5963) fifty-eight (0.9683) first (0.9465) floor (0.8029) letter (0.7869) e (0.4361)]
(15) *Saplen*: I need to confirm your address. Did you say Andalucia street, number 58, first floor letter e? Please answer yes or no.
(16) *Simulated user*: Yes [Recognition hypothesis: yes (0.5639)]
(17) *Saplen*: OK. Address Andalucia street, number 58, first floor letter e. Would you like to drink something?
(18) *Simulated user*: A large beer [Recognition hypothesis: a (0.5698) large (0.2631) beer (0.9764)]
(19) *Saplen*: OK. One beer. Which size would you like?
(20) *Simulated user*: Large [Recognition hypothesis: large (0.5969)]
(21) *Saplen*: OK. So that's one large beer. Would you like anything else?
(22) *Simulated user*: No [Recognition hypothesis: no (0.5972)]
(23) *Saplen*: I need to confirm your order. I think you ordered three ham sandwiches and one large beer. Is this correct? Please answer yes or no.
(24) *Simulated user*: Yes [Recognition hypothesis: yes (0.6384)]
(25) *Saplen*: OK. The price is 12 euros and ten cents. Is this OK for you? Please answer yes or no.
(26) *Simulated user*: Yes [Recognition hypothesis: yes (0.5821)]
(27) *Saplen*: OK. The delivery service will take 30 minutes approximately to get to your address. Is this OK for you? Please answer yes or no.
(28) *Simulated user*: Yes [Recognition hypothesis: yes (0.5690)]
(29) *Saplen*: OK. Thank you very much for using our telephone service automatic system. Good night.

Dialogue 1. Dialogue with a very cooperative user.

This sample dialogue shows that the simulated user always behaved as expected, answering each system prompt with the required type of utterance, which is the modelled behaviour for the very cooperative users. Dialogue turns (4), (10), (16), (24), (26) and (28) show that the simulated user answered the explicit confirmation prompts generated by the system with "yes/no" responses. Dialogue turn (2) was used to enter a product order in response to turn

(1), turn (8) was used to enter a telephone number is response to turn (7), turn (12) was used to enter a postal code in response to turn (11), and turn (14) was employed to enter the address in response to turn (13). The system employed turns (7), (11), (13) and (17) to make implicit confirmations of data previously obtained. These were taken as confirmed by the simulated user as it did not make any correction.

4.1 Frames used

The semantic analyser of the Saplen system creates one (or more) frames from each recognition hypothesis provided by the speech recogniser. The frames generated by the semantic analyser of the system have a variable number of slots, depending on the utterance types from which they are created. For example, those created from confirmations, telephone numbers and postal codes contain only one slot, while those created from food orders contain up to five, and those created from drink orders and addresses contain up to six slots. We consider that some frame slots are "obligatory" as they must be filled in with a value (e.g. the amount in a food order), while others are "optional" and can be left empty (e.g. the temperature in a drinks order). The frames impose restrictions concerning valid values for the slots; for example, a frame used to store a telephone number contains just one slot that has a valid value if it contains nine digits.

Two types of frames are distinguished in this paper: reference frames and obtained frames. The first type corresponds to frames created beforehand and associated with the utterances in the dialogue corpus (discussed in Sect. 3.1). For example: (<PHONE_NUMBER> = "958123456") is a reference frame associated with the utterance "My telephone number is 9 5 8 1 2 3 4 5 6". The obtained frames are created by the semantic analyser of the Saplen system as it processes the recognition hypotheses, and thus they can be affected by speech recognition errors. The semantic analyser assigns a confidence score to each frame slot taking into account the confidence scores of the words in the slot (discussed in Sect. 3). If the slot only contains one word, the confidence score of the slot is that of the word. If the slot contains several words (e.g. a telephone number), its confidence score is the lowest confidence score of the words in the set of words. By employing this policy the dialogue system confirms the content of the slot if there is low confidence in the recognition of at least one word in the slot. These scores are used by the dialogue manager of the system to either accept the data in the slots or to generate confirmation prompts. If an obtained frame is exactly the same as the reference frame associated with the analysed utterance, the obtained frame is said to be correct; otherwise we say it is incorrect. All the reference frames are correct since they were created from the orthographic transcriptions of the utterances in the corpus, which are obviously not affected by speech recognition errors.

5 The proposed technique

The technique proposed in this paper for artificially simulating the behaviour of different types of user is an improvement on the system discussed in the previous section, which only allows simulation of the behaviour of users who always answer the system prompts with the required types of utterance (as observed in Dialogue 1). The proposed technique overcomes this limitation by enabling the simulation of three different degrees of user cooperativeness: *very cooperative*, *cooperative* and *not very cooperative*. Therefore this is useful for more exhaustive testing of the performance of a dialogue system in order to identify possible design or performance problems.

5.1 Differences and similarities with other simulated users

The simulated user we propose has several similarities with previous studies of user simulation with dialogue systems. It is related to the work presented by Scheffler and Young (2000, 2001) in which the intentions of the user being simulated are represented as goals. Our proposal also uses goals to represent user intentions, as can be observed in Fig. 2. A difference between both approaches is that for Scheffer and Young (2000, 2001) each goal represents a specification of the dialogue transaction that the user wants to accomplish, while in our approach each goal represents the semantic content of at least one utterance in the corpus.

The proposed simulated user is also related to that presented by Möller et. al (2006), which is able to provide input that is invalid in the current dialogue state but that would be valid in some other dialogue state. In our approach the dialogue state is directly related to the current prompt generated by the dialogue system. As will be explained in Sect. 5.2, when our simulated user models the behaviour of not very cooperative users, it can provide invalid input data for the current system prompt, e.g. an address when the system prompted for a product order. A difference between the two approaches is that our simulated user cannot issue commands or ask for information that is not supported by the dialogue system, which is a functionality supported by the system of Möller et al. (2006). However, this behaviour could be easily implemented, provided that there are recordings in the corpus of the utterances representing those commands.

Another feature of the proposed simulated user is that it is not aimed at modelling the behaviour of both users and dialogue systems, which is the case of the system presented by Cuayáhuitl et al. (2005), among others. These authors compare the automatically generated dialogues with real dialogues in order to determine the realism of the simulated dialogues. On the contrary, our proposal is aimed at just modelling the behaviour of users and the generated dialogues are not compared with real ones, but are used to find out problems in the design or performance of particular components of the dialogue system. In the experiments presented in Section 6 the simulated user was used to test the performance of the speech recogniser, semantic analyser and dialogue manager of the Saplen system.

5.2 Simulation of different types of user

As commented above, the proposed simulated user models the behaviour of three kinds of user, which in this paper are called *very cooperative*, *cooperative* and *not very cooperative*. To simulate the behaviour of the very cooperative users the system always generates as response the kind of utterance requested by the dialogue system, as can be observed in Dialogue 1. The advantage of this kind of simulation is that it makes it possible to have an estimate of the system performance interacting with "optimal" users who are able to provide exactly the data requested by the system in each interaction.

To simulate the behaviour of cooperative users the simulated user does not always respond with the kind of utterance requested by the dialogue system, but always generates responses that are appropriate for the system prompts. For example, it can respond "Three" for turn (3) in Dialogue 1. This can be useful to model experienced users who know how to make the interaction go faster by providing in advance the data that the system will require in subsequent prompts, e.g. uttering directly "Three" instead of "No" in turn (3) of Dialogue 1. Simulating this kind of user makes it possible to evaluate the ability and "intelligence" of the system to deal appropriately with responses that do not match every system prompt exactly.

Table 1 Strategy to generate responses for the not very cooperative users

System prompt type	Response type to be generated	User action (or user type)
Product order?	Product order	Providing the requested data
Product order?	Telephone number	Deliberately uncooperative, inexperienced or nervous
Product order?	Postal code	Deliberately uncooperative, inexperienced or nervous
Product order?	User address	Deliberately uncooperative, inexperienced or nervous
Telephone number?	Product order	Correcting system misunderstanding
Telephone number?	Telephone number	Providing the requested data
Telephone number?	Postal code	Deliberately uncooperative, inexperienced or nervous
Telephone number?	User address	Deliberately uncooperative, inexperienced or nervous
Postal code?	Product order	Deliberately uncooperative, inexperienced or nervous
Postal code?	Telephone number	Correcting system misunderstanding
Postal code?	Postal code	Providing the requested data
Postal code?	User address	Deliberately uncooperative, inexperienced or nervous
Address?	Product order	Deliberately uncooperative, inexperienced or nervous
Address?	Telephone number	Deliberately uncooperative, inexperienced or nervous
Address?	Postal code	Correcting system misunderstanding
Address?	User address	Providing the requested data

To simulate the behaviour of not very cooperative users the simulated user generates appropriate responses for some system prompts, but for others generates responses that are completely inappropriate. For example, it can generate an address when the system prompted for a food order. The advantage of this kind of simulation is that it makes it possible to model the behaviour of users that perhaps call the system just to try to make it fail, and also of users who because of inexperience or being nervous may answer some prompts with unexpected utterances. Therefore, this type of simulation allows us to test the robustness of the speech recogniser and semantic analyser to deal with these utterances. Also, it can be useful to test the ability of the dialogue manager to appropriately handle data items that are not expected. For example, if the address was provided unexpectedly in advance, the dialogue manager should not prompt for these data later as they were already provided. For modelling the not very cooperative users in our application domain we considered the four types of utterance in the scenarios (product orders, telephone numbers, postal codes and addresses) and made the simulated user generate randomly any of these kinds of utterance in response to the system prompts to enter product orders, telephone number, postal code and address. This strategy of generating a random response is shown in Table 1. The procedure to generate responses for the prompt types not shown in the table is the same as that for the cooperative users.

The table shows that not very cooperative users can sometimes behave "cooperatively" by answering the system prompts with the expected utterance types, and also by correcting system misunderstandings when necessary. These users can also respond with utterances that are not expected to try to make the system fail deliberately, or because of inexperience or nervousness when using the system.

5.3 Implementation of the simulated user

The proposed simulated user is a rule-based automaton (Sect. 2.2) that receives as input the current prompt generated by the dialogue system and the semantic representation that the system obtained ($\mathsf{SR}_{\mathrm{obt}}$) from the analysis of the previous simulated user response. The output is a voice sample file which is the next simulated user response, representing a response uttered by a real user. This response is the next input for the dialogue system. For purposes of illustration Fig. 3 shows algorithmically part of the simulated user employed to test the Saplen system used in the experiments.

The algorithm uses the SR variable to store the semantic representation associated with the voice signal file to be used in the simulated user's next response. The value for this variable is decided in parts 1–4 of the algorithm and is stored in the $\mathsf{SR}_{\mathrm{cor}}$ variable, which represents the correct semantic representation associated with the simulated user response. The remainder of this section explains how the algorithm generates responses for implicit and explicit system confirmation prompts, handles scenario goals, deals with system prompts to get missing data, selects utterances from the corpus and makes some preparations for the next interaction.

5.4 Generation of responses for system implicit confirmations

The first part of the algorithm deals with the generation of responses to answer implicit confirmation prompts generated by the system such as "OK. So that's three ham sandwiches. Please say your telephone number". For this kind of prompt the simulated user provides the requested data if it does not detect a misunderstanding of its previous response. Otherwise it generates a response to let the system know that there was a mistake (e.g. "It is incorrect"). This kind of response for mistakes, noted as ERROR_INDICATION in the algorithm, is generated if the following two conditions are satisfied:

(i) The obtained semantic representation ($\mathsf{SR}_{\mathrm{obt}}$) is different from the correct semantic representation ($\mathsf{SR}_{\mathrm{cor}}$). For example, this happens if the previous response was "Three ham sandwiches" but the system understood "Six ham sandwiches". The correction is necessary since otherwise the system would consider a misunderstood order as being implicitly confirmed by the simulated user.

(ii) The obtained semantic representation is not a subset of the correct semantic representation used as a reference. For example, it is a subset if the simulated user responded "Three ham sandwiches" but the system understood "Ham sandwiches". In this case the simulated user does not make a correction since the dialogue system will try to get missing data, e.g. generating the prompt: "OK. So that's ham sandwiches. How many do you want?".

5.5 Generation of responses for system explicit confirmations

The second part of the algorithm is concerned with the generation of responses for explicit confirmations generated by the dialogue system such as "I have to confirm your order. Did you say you want six ham sandwiches? Please say yes or no". For this kind of prompt the simulated user generates a "yes/no" response if it is modelling a very cooperative user. If it is simulating a cooperative or a not very cooperative user it generates a response containing the data item that the system is trying to confirm (e.g. "Three"). For reasons of space, Fig. 3 only shows two such prompt types but our complete implementation contains similar actions for the 30 explicit confirmation prompts that the Saplen system can generate.

```
simulated_user_response_generation (Inputs: prompt, $SR_{obt}$; Output: voice_signal_file)
   cooperativeness = 1 /* '1' is for very cooperative users. Other possible values are '2' or '3' */
/*  SR initialization */
   SR = ""
/*  Part 1: implicit confirmations */
   if  ( (prompt is not an explicit confirmation) and (not first simulator response) ) then {
     if ( ($SR_{obt}$ != $SR_{cor}$) and ($SR_{obt}$ not included in $SR_{cor}$) ) then SR = ERROR_INDICATION }
/* Part 2: explicit confirmations */
   if (SR = "") {
      if (prompt = TELEPHONE_OK?) then {
                 if ($SR_{obt}$ = $SR_{cor}$) then { if (cooperativeness = 1) then SR = AFFIRMAT_CONF
                                             else SR = TELEPHONE_NUMBER }
                                    else  { if (cooperativeness = 1) then SR = NEGAT_CONF
                                             else SR = TELEPHONE_NUMBER }
      else
      ...
      if (prompt = ORDERS_OK?) then {check_ordered_products()
                                     if (ok) then SR = AFFIRMAT_CONF
                                            else SR = NEGAT_CONF }
   }
/* Part 3: management of scenario goals */
   if (SR = "") {
       if (prompt = SOMETHING_TO_EAT?) then {
          if ( (cooperativeness = 1) or (cooperativeness = 2) ) then search_scenario_for_food_order(goal)
                                                                 else search_scenario_for_goal(goal)
           SR = goal
          mark_as_used(goal)
          create_data_items(goal) }
      else
       if (prompt = TELEPHONE_NUMBER?) then {
          if ( (cooperativeness = 1) or (cooperativeness = 2) ) then
search_scenario_for_telephone_number(goal)
                                                                 else search_scenario_for_goal(goal)
          SR = goal
          mark_as_used(goal)
          create_data_items(goal) }
      ...
   }
/* Part 4: management of missing data items */
   if (SR = "") {
      if (prompt = AMOUNT?) then SR = AMOUNT_ITEM
      else
      if (prompt = TASTE?) then SR = TASTE_ITEM
      …
   }
/* Part 5: response generation */
   look_for_file(SR, ortographic_transcription.txt, voice_signal.wav)
   include_in_log_file(prompt, ortographic_transcription.txt)
   voice_signal_file = voice_signal.wav
/* Part 6: dialogue history storage */
   store_in_information_storage_of_simulated_user($SR_{obt}$)
/* Part 7: setting correct semantic representation as the desired semantic representation */
   $SR_{cor}$ = SR
End
```

Fig. 3 Procedure to generate a response from the simulated user

The first prompt shown is concerned with the confirmation of the telephone number, and the processing is analogous for the confirmation of postal code, address and product orders. This kind of processing could be easily adapted to implement a simulated user for a different application domain by simply changing the prompt names. The second prompt is concerned with the confirmation of the set of product orders understood. This is an example of domain-dependent processing which uses a specific function to compare the set of product orders understood by the system with that in the scenario employed. This function may not be useful for simulated users interacting with dialogue systems designed for other application domains. Hence the system designers should implement the necessary functions in their domains, e.g. to decide whether the dialogue system correctly understood the details of a flight booking.

5.6 Management of scenario goals

The third part of the algorithm is concerned with the management of scenario goals. Its objective is to find the semantic representation (SR variable) associated with each goal in the scenario employed. To save space Fig. 3 just shows the processing for two system prompts but our complete implementation includes the processing of the 53 different prompts that the Saplen system can generate. Each scenario employed (see Fig. 2) contains five product orders, one telephone number, one postal code and one address. Therefore when the system prompts for food orders (e.g. "Would you like to have anything to eat?") the algorithm searches for the semantic representation of a food order in the scenario that has not been used already. This representation is the next goal to be achieved by the system. The algorithm also creates the sub-goals corresponding to the goal (amount, type, ingredients, etc.), which are necessary to recover from possible speech recognition deletion errors (Rabiner and Juang, 1993; Huang et al. 2001), as will be discussed in the following section. For example, the two sub-goals for the "ham sandwich" order in the sample scenario shown in Fig. 2 would be (<AMOUNT> = "1") and (<INGREDIENTS> = "HAM").

5.7 Generation of responses to provide missing data

The fourth part of the algorithm deals with the management of system prompts to get missing data. For example, in our application domain data can be missing if the user did not provide all the data required to make a product order or query (e.g. "I want ... uhm ... ham sandwiches"), or did not provide all the necessary details in his address. Data can also be missing because of speech recognition errors that delete words which are necessary to obtain the correct semantic representation, e.g. taste in a drink order, ingredients in a food order or street name in an address. In the interaction with the proposed simulated user data can only be missing because of speech recognition errors, given that all the utterances in the utterance corpus employed to generate responses contain all the data items required.

If after the analysis of a user response the Saplen system detects that data is missing then it generates prompts to obtain these data from the user, as for example "How many ham sandwiches do you want?" To generate responses for these kinds of prompt the simulated user employs the sub-goals created when the current goal was selected from the scenario, as discussed in the previous section. To save space Fig. 3 shows only the processing for two such prompts but our complete implementation includes the processing for the 12 prompt types that the Saplen system can generate to get missing data. The objective is to assign the appropriate value to the SR variable, which represents the semantic representation of the simulated user's next response. For instance, if the system prompts to get the amount of a product order, the algorithm assigns to the SR variable the amount sub-goal of the current goal, e.g. (<AMOUNT> = "1").

5.8 Selection of utterances from the corpus

At this point the algorithm has already decided the kind of utterance to be generated as the simulated user's next response, which is represented as a semantic representation stored in the SR variable. In part 5 it simply looks[2] for an utterance in the corpus (wav file in our implementation) which has an associated semantic representation that matches the SR. This file is the output of the algorithm and the next input for the dialogue system. The current prompt of the dialogue system and the orthographic transcription of this utterance are included in a log file created for posterior evaluation purposes.

5.9 Preparation for the next interaction

Part 6 stores the last semantic representation obtained by the system ($\mathrm{SR}_{\mathrm{obt}}$) into the Information Storage module of the simulated user, just as a human being "stores" in his memory what he thinks the other partner in the conversation has understood. This is necessary because the simulated user's responses for some system prompts depend on the dialogue history. For example, in our application domain the simulated user needs to know all the product orders understood by the system up to a particular moment of the conversation, which enables it to generate the appropriate "yes/no" response for the prompt "OK, would you like to order anything else?" If all the product orders in the employed scenario have been understood by the system, the simulated user generates a negative response and otherwise it generates an affirmative response.

Finally part 7 of the algorithm sets the correct semantic representation ($\mathrm{SR}_{\mathrm{cor}}$) to be the desired semantic representation (SR). This is necessary to compare this representation in the next interaction against that obtained by the dialogue system ($\mathrm{SR}_{\mathrm{obt}}$) and to generate appropriately the next simulated user response.

5.10 Example of a dialogue with a not very cooperative user

This section shows a sample generated dialogue (translated from Spanish into English) with the simulated user modelling the behaviour of a not very cooperative user and the system employing the PILM (prompt-independent language model).

(1) *Saplen*: Good evening, welcome to our restaurant. This is our automatic telephone service system. The minimum price for the home delivery service is six euros. Please speak after the tone. What would you like to have?

(2) *Simulated user*: Acera de Canasteros street, number one, third floor, apartment letter e [Recognition hypothesis: cherry (1.0000) make (0.5997) an (0.0718) order (0.718) of (0.6070) elm (0.4084)]

(3) *Saplen*: I'm sorry, I didn't understand. You must enter product orders one by one. For example, if you want to order a sandwich and a beer, order first the sandwich and then the beer. What would you like to have?

(4) *Simulated user*: One large orange fanta [Recognition hypothesis: one (0.7991) large (1.0000) orange (1.0000) fanta (0.5997)]

---

[2] To employ all the utterances available in the corpus the search procedure first selects utterances that have not been used before. When all utterances have been used it selects those least used, which helps to employ all the possible utterances approximately the same number of times.

(5) *Saplen*: OK. So that's one large orange fanta. Please say your telephone number.
(6) *Simulated user*: One large cola [Recognition hypothesis: one (0.5044) large (0.8999) cola (0.7001)]
(7) *Saplen*: OK. So that's one large cola. Please say your telephone number.
(8) *Simulated user*: 1 8 0 0 1 [Recognition hypothesis: one (1.00000) eight (1.0000) zero (1.0000) zero (1.0000) one (0.7998)]
(9) *Saplen*: OK. Telephone number 1, 8, 0, 0, 1. Please say your postal code.
(10) *Simulated user*: It is wrong [Recognition hypothesis: it (0.7032) is (0.6996) wrong (1.0000)]
(11) *Saplen*: Do you want to change the telephone number? Please say yes or no.
(12) *Simulated user*: Yes [Recognition hypothesis: yes (0.7032)]
(13) *Saplen*: OK. Please say your telephone number again.
(14) *Simulated user*: One ham sandwich [Recognition hypothesis: I (0.3098) want (0.4023) a (0.5021) ham (0.8987) sandwich (0.6572)]
(15) *Saplen*: OK. So that's one ham sandwich. Please say your telephone number again.
(16) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.8986) five (0.5621) eight (0.9703) one (0.5499) two (0.5890) two (0.7861) three (0.4581) four (0.6548) five (0.3860)]
(17) *Saplen*: I need to confirm your telephone number. Did you say 9, 5, 8, 1, 2, 2, 3, 4, 5? Please answer yes or no.
(18) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.9676) five (0.8510) eight (0.8701) one (0.6021) two (0.6719) two (0.6123) six (0.5081) four (0.8439) five (0.6219)]
(19) *Saplen*: OK. Telephone number 9, 5, 8, 1, 2, 2, 6, 4, 5. Please say your postal code
(20) *Simulated user*: You are wrong [Recognition hypothesis: you (0.4048) are (0.6992) wrong (1.00000)]
(21) *Saplen*: Do you want to change the telephone number? Please say yes or no.
(22) *Simulated user*: Yes [Recognition hypothesis: yes (0.5223)]
(23) *Saplen*: OK. Please say your telephone number again.
(24) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.9439) five (0.9437) eight (0.6491) one (0.7320) two (0.7328) two (0.7412) three (0.4981) four (0.7429) five (0.9812)]
(25) *Saplen*: I have to confirm your telephone number again. Did you say 9, 5, 8, 1, 2, 2, 3, 4, 5? Please answer yes or no.
(26) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.6742) five (0.8361) eight (0.8531) one (0.5832) two (0.9531) two (0.9671) six (0.9651) four (0.8540) five (0.8512)]
(27) *Saplen*: OK. Telephone number 9, 5, 8, 1, 2, 2, 3, 4, 5. Please say your postal code.
(28) *Simulated user*: 1 8 0 0 1 [Recognition hypothesis: one (0.7984) eight (0.7941) zero (1.0000) zero (1.0000) one (0.5048)]
(29) *Saplen*: OK. Postal code number 1, 8, 0, 0, 1. Please say your address, as for example, Elm street, number 13, first floor, letter e
(30) *Simulated user*: Acera de Canasteros street, number one, third floor, apartment letter e [Recognition hypothesis: acera-de-canasteros (1.0000) street (1.000) zero (0.9046) zero (0.9046) twelve (1.0000) beer (0.2862)]

*** INTERACTION LIMIT (30) REACHED, DIALOGUE CANCELLED BY SIMULATED USER ***

Dialogue 2. Dialogue with a not very cooperative user.

If we compare this dialogue with Dialogue 1 we can observe the different behaviour of the simulated user as well as the difficulties for the system to obtain the necessary data and to correct understanding errors. Dialogue turns (2), (4), (6), (8) and (14) show the random selection of utterances to answer system prompts to enter product orders, telephone numbers, postal codes and addresses (discussed in Sect. 5.2). Turns (18) and (26) show that the dialogue system repeated the data being confirmed by the system (telephone number) instead of uttering the "yes/no" requested response, which is the modelled behaviour for cooperative and not very cooperative users. Given the difficulties in interacting with the system, the simulated user cancelled the dialogue as the interaction limit set to 30 turns in total was reached, as will be discussed in Section 6.1.

6 Experiments

The goal of the experiments was to employ the proposed technique to test the performance of the Saplen system and improve it by identifying problems in the performance of its speech recogniser, semantic analyser or dialogue manager. By fixing these problems the system could be more robust to deal with a variety of users in a variety of interaction modes.

6.1 Evaluation measures

The evaluation was carried out in terms of word accuracy (WA), sentence understanding (SU) and task completion (TC). WA is the proportion of correctly recognised words. It was computed as $\mathrm{WA} = (w_t - w_i - w_s - w_d) \times 100/w_t$, where w_t is the total number of words in the analysed utterances, and w_i, w_s and w_d are the number of words inserted, substituted and deleted by the speech recogniser, respectively.

SU is the proportion of utterances correctly understood by the system when it interacts with the simulated user. This was computed as $\mathrm{SU} = S_u \times 100/S_t$, where S_u is the number of analysed utterances for which the semantic representations (frames) obtained were correct (i.e. $SR_{obt} = SR_{cor}$) and S_t is the total number of analysed utterances. If an utterance was partially understood, for example the order "one large orange fanta" was understood as "one large fanta" we did not consider it to be SU, as there was not an exact match between the obtained and the correct semantic representation.

TC is the proportion of successful dialogues, i.e. the percentage of dialogues that ended with all the scenario goals achieved by the simulated user. This was computed as $\mathrm{TC} = D_c \times 100/D_t$, where D_c is the number of successful dialogues and D_t is the total number of dialogues. In order to avoid excessively long dialogues between the system and the simulated user, which would not be accepted by real users, we made the latter cancel the interaction with the system if the total number of interactions (i.e. of system plus simulated user turns) exceeded a threshold that we set to 30 interactions. The reason for using this maximum limit is that, taking into account the structure of the scenarios[3] as well as the dialogue management strategy of the Saplen system, a dialogue without any error correction requires

---

[3] All the scenarios employed in the experiments included five product orders, a telephone number, a postal code and an address.

Table 2 Utterances used for training and test

Utterance type	No. training utterances	No. testing utterances
Product order	250	250
Telephone number	250	250
Postal code	250	250
Address	250	250
Query	125	125
Confirmation	125	125
Amount	125	125
Food name	125	125
Ingredient	125	125
Drink name	125	125
Size	125	125
Taste	125	125
Temperature	125	125
Street name	125	125
Building number	125	125
Building floor	125	125
Apartment letter	125	125
Error indication	125	125
Total	2,750	2,750

20 turns in total to make the orders, provide the user data (postal code, telephone number, and address) and answer the confirmation prompts generated by the system. Considering 30 as an interaction limit means that we permitted $30 - 20 = 10$ correction turns per dialogue, which we consider an ample margin for the simulated user to correct possible errors. In other words, the three kinds of modelled user (very cooperative, cooperative and not very cooperative) were supposed to be patient enough to accept employing 10 dialogue turns to correct system errors before hanging up. Cancelled dialogues were not considered successful and thus decreased the TC rate.

6.2 Utterance corpus

To carry out the experiments we employed two utterance corpora, one for training and the other for testing, that we have used in previous studies (López-Cózar et al. 2003; López-Cózar and Callejas 2005). We ensured that no training utterances were included in the testing corpus. Both corpora were created employing the dialogue corpus discussed in Sect. 3.1, in which we selected 5,500 client utterances at random among the 18 utterance types shown in Table 2.

Both corpora include the orthographic transcriptions of the utterances as well as their corresponding reference frames. One half of the utterances that the simulated user employed to correct system errors and to confirm data were used for training and the other half were used for testing. These utterances were not used as scenario goals given that they are scenario-independent.

6.3 Language modelling for speech recognition

The Saplen system was configured to use two different kinds of language model for speech recognition: one based on 17 prompt-dependent language models (PDLMs), in the form of word bigrams (Rabiner and Juang 1993), whilst the other was based on one prompt-independent language model (PILM), also a word bigram. Both kinds of language model have been used in previous studies (López-Cózar et al. 2003; López-Cózar and Callejas 2005). Our goal was to test the performance of the Saplen system interacting with the three kinds of modelled users employing both kinds of language model for each kind of user.

As shown in (López-Cózar et al. 2003) the PDLMs provide good results if the users answer each system prompt with the expected kind of utterance (e.g. a telephone number when the system prompted for a telephone number). This happens because each utterance is analysed using a bigram trained with utterances of the same type as the utterance being analysed. To create this language model we compiled a specific word bigram from the orthographic transcriptions of the training utterances of each type shown in Table 2 excluding the error indications[4], which amounts to 17 bigrams in total. The 125 error indication utterances used for training were included in each utterance set to compile the 17 bigrams, which allows the recognition of error indications whenever the speech recogniser employs any of these bigrams. The problem with the PDLMs is that these provide very poor results if the users respond to system prompts with a type of utterance that does not match the active grammar (e.g. an address when the system prompted for a telephone number). This happens because the utterances are analysed employing a grammar compiled from utterances of a different type. Therefore, this language model is not appropriate if we want to provide users with a natural interaction that enables them to answer any system prompt with any kind of utterance within the application domain, which is something that they would probably do when interacting with a real operator. Contrary to what happens with the PDLMs, the PILM permits the recognition of any kind of utterance within the domain, which helps to provide users with a natural interaction. However, the accuracy is in general lower than with the PDLMs given that the vocabulary is much larger and there are many more types of utterance to be considered in the analysis of each utterance.

6.4 Scenarios

To automatically generate dialogues between the Saplen system and the proposed simulated user we designed 50 scenarios similar to the one shown in Fig. 2. The scenario goals (frames) were selected by choosing utterances at random in the test utterance corpus (discussed in Sect. 6.2) corresponding to product orders, telephone numbers, postal codes and addresses. We ensured that for each scenario goal there was at least one utterance in the test corpus associated with the frame, since otherwise it would be impossible for the simulated user to generate a response to some system prompts. For example, the scenario goal: (<PHONE_NUMBER> = "958275360") lets the simulated user answer the prompt: "Please say your telephone number". Thus at least one utterance with this reference frame was necessary in the test corpus. We also ensured that at least one utterance associated with each scenario sub-goal was in the test corpus to allow the simulated user to answer error-correction prompts generated by the system. For example, the scenario shown in Fig. 2 would require utterances in the test corpus which are associated with the following

---

[4] These utterances (e.g. "You made a mistake", "It is incorrect") are uttered by the user to correct errors made by the system. When one of these is uttered the system prompts the user to confirm the change of the data obtained from the last user interaction.

Table 3 Saplen system average performance (in %)

	PDLMs			PILM		
	WA	SU	TC	WA	SU	TC
Very cooperative users	90.05	85.18	70.56	75.4	57.86	11.67
Cooperative users	70.56	70.76	21.67	76.6	55.71	5.56
Not very cooperative users	43.69	56.82	11.13	77.87	53.28	4.47

reference frames: (<AMOUNT> = "1"), (<INGREDIENTS> = "HAM"), (<INGREDIENTS> = "CHEESE"), (<SIZE> = "LARGE"), (<TASTE> = "CHOCOLATE") and (<ADDR_NAME> = "ANDALUCÍA"), among others. These frames would allow the simulated user to answer e.g. the error-correction prompts: "How many cheese sandwiches did you say?" and "Please say again the name of the street".

6.5 Results and discussion

Using the proposed technique we generated 20 dialogues per scenario according to the kind of language model (PDLMs and PILM) and user type modelled (very cooperative, cooperative and not very cooperative). This makes a total of $20 \times 50 \times 2 \times 3 = 6{,}000$ dialogues between the Saplen system and the simulated user. Table 3 shows the average results obtained in terms of WA, SU and TC.

The table shows that the best performance was achieved for the very cooperative users regardless of the language model employed. The differences in the performance are more clearly observed when the PDLMs were employed. The reason is that when the simulated user modelled the behaviour of the very cooperative users it always provided responses that matched the current system prompt. Therefore using the PDLMs each utterance was analysed employing the appropriate recognition grammar. The scores decrease for the cooperative and the not very cooperative users, given that when the simulated user modelled these kinds of user, it sometimes provided utterances that did not match the current system prompt. For the cooperative users this happened only for the confirmations but for the not very cooperative users it happened also for prompts to enter product orders, telephone numbers, postal codes and addresses. Therefore in both cases there were utterances that were analysed employing inappropriate recognition grammars, especially in the case of the not very cooperative users. According to these results it can be said that the system should only employ the PDLMs if the real users were very cooperative, since otherwise the performance would be very poor.

When the PILM was employed the values for the evaluation measures were very similar for the three kinds of user. The reason is that regardless of user type, the simulated user responses always matched the recognition grammar, as it was compiled from training utterances permitted for all the system prompts. It is interesting to note that, contrary to what happened with the PDLMs, the WA score achieved for the not very cooperative users (77.87%) was slightly higher than that obtained for the very cooperative users (75.4%). However the trend is different for the SU: the rate achieved for the very cooperative users (57.86%) was slightly higher than that obtained for the not very cooperative users (53.28%). The reason for this apparent contradiction is that when the simulated user modelled the behaviour of the very cooperative users, the answers to confirmation prompts were "yes/no" responses, whereas in the case of cooperative and not very cooperative users these were repetitions of the data being confirmed by the system. Hence, when the data was a postal code or a telephone number the simulated user repeated the five (postal code) or nine (telephone number) digits. We observed that, especially for telephone numbers, at least one digit of the sequence was in some cases misrecognised or recognised with low confidence, but the remaining digits were correctly recognised. Therefore, given that each digit is considered as a word for the WA computation, employing telephone numbers for the confirmation of telephone numbers increased the overall proportion of correctly recognised words. This effect was more apparent for not very cooperative users than for the cooperative users (WA values 77.87% vs. 76.6%) because for the former the random selection of responses caused more telephone numbers and postal codes to be generated as responses, thus increasing to a greater extent the proportion of correctly recognised words. Nevertheless what is really important to have as a measure of acceptable system performance is a high rate of SU. When the PILM was employed the SU was very low for the three kinds of user (lower than 60%) which made the system performance totally unacceptable given that TC was too low in the three cases (only 11.67% in the best case).

As discussed at the beginning of Sect. 6, a goal of the experiments was to identify problems with the speech recogniser, semantic analyser and dialogue manager of the Saplen system, to fix those and thus increase the system's robustness to deal with a variety of users. To do this we focused on the generated dialogues with very low values for the evaluation measures and analysed these to find the reasons for the low system performance. The remainder of this section discusses the information obtained from the analysis.

6.6 Performance with the very cooperative users

When the PDLMs were employed the utterances that the simulated user generated as responses always matched the active speech recognition grammars, which caused WA to be quite high (90.05%). The 10% word error rate was caused by three factors. One is that some "yes/no" answers to confirmation prompts were misrecognised, e.g. the word "sí" (yes) was sometimes substituted by the word "te". Another reason is that there were problems recognising some addresses for which not all data items were recognised. The third reason is that there were many recognition errors if the speakers had strong southern Spanish accents, as they usually do not pronounce the final 's' of plural words. For example, when they uttered the words "bocadillos" (sandwiches) and "grandes" (large) the recognition hypotheses were "bocadillo" (sandwich) and "grande" (large). Given that these errors in the number correspondence do not affect the semantics of the utterances, most of the product orders were correctly understood even though some words were incorrectly recognised. The average TC employing the PDLMs was 70.56% which suggests that the system performance can be considered more or less acceptable for real users who may behave similarly to the very cooperative users modelled in this study.

When the PILM was employed there were also many speech recognition errors in the responses to system confirmation prompts, especially if these were uttered by speakers with strong southern Spanish accents. Given that these users omit the final 's' of plural words, as discussed above, because of acoustic similarity the word "no" was often substituted by the word "dos" (two), "uno" (one) or "error", while the word "sí" (yes) was often substituted by the word "seis" (six). Moreover in many cases the words "yes" and "no" were discarded by the semantic analyser of the system as their confidence scores (discussed in Sect. 3) were smaller than the lower confidence threshold employed (set to 0.3). Because of these problems there were many repetitive confirmation turns to get data confirmed, which lengthened the dialogues and caused some of these to be cancelled as the interaction limit (30 turns) was reached before all the scenario goals were achieved. These problems are illustrated in the following sample dialogue.

(1) *Saplen*: Please say your telephone number.
(2) *Simulated user*: 9 5 8 1 2 3 4 5 6 [Recognition hypothesis: nine (0.8463) five (0.8353) eight (0.7438) one (0.6279) two (0.9583) six (0.4982) four (0.6378) five (0.7391) six (0.8537)]
(3) *Saplen*: I have to confirm your telephone number. Did you say 9, 5, 8, 1, 2, 6, 4, 5, 6? Please say yes or no.
(4) *Simulated user*: No [Recognition hypothesis: no (0.1206)]
(5) *Saplen*: I'm sorry, I didn't understand. I have to confirm your telephone number again. Did you say 9, 5, 8, 1, 2, 6, 4, 5, 6? Please say yes or no.
(6) *Simulated user*: No [Recognition hypothesis: error (0.5216)]
(7) *Saplen*: Do you want to change the telephone number? Please say yes or no.
(8) *Simulated user*: Yes [Recognition hypothesis: yes (0.5928)]
(9) *Saplen*: OK. Please say your telephone number again.
(10) *Simulated user*: 9 5 8 1 2 3 4 5 6 [Recognition hypothesis: nine (0.8632) five (0.6458) eight (0.5367) one (0.2581) two (0.8632) six (0.2365) four (0.3625) five (0.6548) six (0.4862)]
(11) *Saplen*: OK. I have to confirm your telephone number again. Did you say 9, 5, 8, 1, 2, 6, 4, 5, 6? Please say yes or no.
(12) *Simulated user*: No [Recognition hypothesis: two (0.5642)]
(13) *Saplen*: I have to confirm your telephone number again. Did you say 2? Please say yes or no.
(14) *Simulated user*: No [Recognition hypothesis: two (0.4642)]
(15) *Saplen*: I have to confirm your telephone number again. Did you say 2? Please say yes or no.
(16) *Simulated user*: No [Recognition hypothesis: one (0.3624)]
(17) *Saplen*: I have to confirm your telephone number again. Did you say 1? Please say yes or no.

Dialogue 3. Problems in the confirmation strategy for the very cooperative users employing the PILM.

We also found out from the analysis of the dialogues that there were problems in the understanding of some addresses, given that some data items were correctly recognised (e.g. street name) while others were incorrectly recognised or missing (e.g. building number). Because of these problems the system needed to initiate repair sub-dialogues to prompt for the missing data and confirm the obtained data if recognised with low confidence. Again, these additional turns increased the total number of dialogue turns, and thus some dialogues had to be cancelled as the interaction limit was reached. As a consequence of all these problems the average TC employing the PILM was 11.67%, which is too low to consider the system performance acceptable for real users who may behave similarly to the very cooperative users modelled in this study.

6.7 Performance with the cooperative users

When the PDMLs were employed the WA for cooperative users was lower than for the not very cooperative users (70.56%). The reason is that in addition to facing the problems discussed in the previous section, in this case the system had to face the problem that the utterances used to answer confirmation prompts were not permitted by the active grammars. As discussed in Sect. 5.2, the cooperative users answer confirmation prompts by repeating the data that the system is trying to confirm, although it always prompts for a "yes/no" response. This behaviour is illustrated in the following sample dialogue.

(1) *Saplen*: Please say your telephone number.
(2) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: seven (0.4025) eight (0.3976) one (0.7002) two (0.6916) seven (0.4698) five (0.7284)]
(3) *Saplen*: I have to confirm your telephone number. Did you say 7, 8, 1, 2, 7, 5? Please say yes or no.
(4) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: no (1.0000) was (1.0000) a (1.0000) mistake (1.0000) a (1.0000) mistake (1.0000) wrong (1.0000)]
(5) *Saplen*: Do you want to change the telephone number? Please say yes or no.
(6) *Simulated user*: Yes [Recognition hypothesis: yes (0.8567)]
(7) *Saplen*: OK. Please say your telephone number again.
(8) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.4025) five (0.7482) eight (0.3976) one (0.7002) two (0.6916) two (0.7202) three (0.7597) four (0.8263) five (0.9529)]
(9) *Saplen*: I have to confirm your telephone number. Did you say 9, 5, 8, 1, 2, 2, 3, 4, 5?. Please say yes or no.
(10) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: you (1.0000) went (0.6008) error (0.6008) yes (0.6007) you (1.0000) went (1.0000) wrong (1.0000)]
(11) *Saplen*: Do you want to change the telephone number? Please say yes or no.
(12) *Simulated user*: Yes [Recognition hypothesis: yes (0.6562)]
(13) *Saplen*: OK. Please say your telephone number again.
(14) *Simulated user*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.7038) five (0.9361) eight (0.8474) one (0.6923) two (0.8931) two (0.8849) three (0.4951) four (0.7935) five (0.9686)]
(15) *Saplen*: OK, telephone number 9, 5, 8, 1, 2, 2, 3, 4, 5. Please enter your postal code.

Dialogue 4. Problems in the confirmation strategy for the cooperative users employing the PDLMs.

The problem identified in the analysis is that the grammar employed to recognise responses to confirmation prompts was initially created considering only users who would utter either a confirmation, a negation or an error indication (i.e. very cooperative users). Because of this, the system confirmation strategy employing the PDLMs failed when it interacted with the modelled cooperative users as their responses were not permitted by the grammar. This problem is easily observed in the recognition hypotheses, which only contain permitted words as shown in turns (4) and (10) of Dialogue 4. On the contrary, product orders, telephone numbers, postal codes and addresses were more or less well understood, although the errors in gender/number correspondences and those for some addresses discussed above also occurred in these dialogues. As a consequence of all the problems the average TC employing the PDLMs was 21.67%, which is obviously too low to consider the system performance acceptable for real users who may behave similarly to the cooperative users modelled in this study.

When the PILM was employed the WA was 76.6%, which is very similar to that obtained for the not very cooperative users (77.87%) employing the same language model. The reason for this low rate is the large amount of errors in the recognition of responses to confirmation prompts (e.g. "no" substituted by "dos", and "sí" substituted by "seis"), and also in gender/number correspondences (e.g. "uno" substituted by "una", and "verdes" (green) substituted by "verde"). As commented above, these errors happened especially when the words were uttered by speakers with strong southern Spanish accents. Analysing the dialogues we observed that, as discussed in Sect. 6.5.1, the affirmative and negative responses were sometimes discarded by the semantic analyser of the system as their confidence scores were smaller than the lower confidence threshold (set to 0.3), provoking the same repetitive confirmation turns. Also the same problems detected for the very cooperative users with the recognition of some addresses were found for the cooperative users, which meant that the system needed to employ extra turns to get and confirm all the data items in the addresses.

In addition we observed a problem in the confirmation strategy that was not observed with the very cooperative users and that was particularly noticeable in the confirmation of telephone numbers. The cooperative simulated user confirms this data by repeating the telephone number instead of generating the "yes/no" response requested by the system. To have a telephone number correctly understood, the Saplen system requires on the one hand that all its digits are recognised with confidence scores greater than the higher confidence threshold (set to 0.5 as discussed in Sect. 3). On the other hand, the system requires an implicit confirmation from the user when it includes the recognised number if the prompt is to get the postal code, as can be observed in turn (7) of Dialogue 5. According to the method employed to assign confidence scores to frame slots as discussed in Sect. 4.1, the confidence score of a slot that contains a telephone number is the lowest confidence score of the digits. For example, the confidence score for the recognition hypothesis "nine (0.5684) five (0.9652) eight (0.5647) one (0.5894) two (0.6954) three (0.9654) three (0.4362) four (0.6584) five (0.5898)" would be 0.4362. Because of all these factors, the system had problems confirming some telephone numbers, especially when these were uttered by speakers with strong southern Spanish accents. The reason is that employing a telephone number to confirm a telephone number tends to require another confirmation, given that it is likely that at least one digit is misrecognised or recognised with low confidence. This problem provoked *in-cascade* dialogue turns that lengthened the dialogues, as can be observed in Dialogue 5, causing some of these to be cancelled as the interaction limit was reached.

(1) *Saplen*: Please say your telephone number.
(2) *Simulated user*: 9 5 8 1 2 3 4 5 6 [Recognition hypothesis: nine (0.6025) five (0.7002) eight (0.6698) one (0.6954) two (0.9532) three (0.4235) four (0.6328) five (0.8654) six (0.6321)]
(3) *Saplen*: I have to confirm your telephone number. Did you say 9, 5, 8, 1, 2, 3, 4, 5, 6?. Please say yes or no.
(4) *Simulated user*: 9 5 8 1 2 3 4 5 6 [Recognition hypothesis: nine (0.9632) five (0.6532) eight (0.6581) one (0.6954) two (0.9564) three (0.6521) four (0.9653) five (0.8524) three (0.4362)]
(5) *Saplen*: I have to confirm your telephone number again. Did you say 9, 5, 8, 1, 2, 3, 4, 5, 3?. Please say yes or no.
(6) *Simulated user*: 9 5 8 1 2 3 4 5 6 [Recognition hypothesis: nine (0.6598) five (0.8632) eight (0.8026) one (0.6354) two (0.7236] three (0.5621) four (0.7635) five (0.9023) six (0.7268)]

(7) *Saplen*: OK. Telephone number 9, 5, 8, 1, 2, 3, 4, 5, 6. Please say your postal code.

Dialogue 5. Inadequate system strategy to confirm telephone numbers.

Because of all these problems the average TC employing the PILM was 5.56%, which is obviously too low to consider the system performance acceptable for real users who may behave similarly to the cooperative users modelled in this study.

6.8 Performance with the not very cooperative users

When the PDLMs were employed the WA was very low (43.69%). One reason for this is the high number of errors in the confirmation turns given that, as discussed in Sect. 5.2, these users do not answer confirmation prompts with "yes/no" responses but repeating the data the system is trying to confirm. Therefore the responses could not be recognised as the utterances permitted by the grammar were only affirmative/negative responses as well as utterances to correct system errors. Another reason is that for these users the simulated user selected at random the kind of utterance to answer system prompts to enter product orders, telephone numbers, postal codes or addresses. For example it could respond with a prompt to enter a telephone number, as can be observed in Dialogue 6. In this sample dialogue the recognition grammar employed was that to recognise telephone numbers, and thus it was not possible to recognise the product order.

(1) *Saplen*: Please say your telephone number.
(2) *User*: one large cola [Recognition hypothesis: one (0.8698) four (0.7698) two (0.2092) four (0.7024) one (0.1219]
(3) *Saplen*: OK. Telephone number 1, 4, 2, 4, 1. Please say your postal code.
(4) *User*: It is wrong [Recognition hypothesis: it (0.5563) is (0.5982) wrong (0.7945)]
(5) *Saplen*: Do you want to change the telephone number? Please say yes or no.
(6) *User*: Yes [Recognition hypothesis: yes (0.8526)]
(7) *Saplen*: OK. Please say your telephone number again.
(8) *User*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.6025) five (0.7002) eight (0.6698) one (0.6954) two (0.9532) two (0.4235) three (0.7941) four (0.6328) five (0.8654)]
(9) *Saplen*: OK. Telephone number 9, 5, 8, 1, 2, 2, 3, 4, 5. Please say your postal code.

Dialogue 6. Utterance misrecognition employing PDLMs.

Because of these problems the average TC employing the PDLMs was 11.13%, which is obviously too low to consider the system performance acceptable for real users who may behave similarly to the not very cooperative users modelled in this study.

When the PILM was employed the value of WA (77.87%) was very similar to that obtained for the other two kinds of modelled user (75.4% and 76.6%) given that in the three cases the same kind of language modelling was employed throughout the whole dialogue regardless of the system prompt. Consequently the SU rate (53.28%) was also similar to that for the other kinds of modelled user (57.97% and 55.71%). As discussed in Sect. 5.2, the behaviour of the not very cooperative users modelled is very similar to that of the cooperative users, with the difference that the former features a random selection of utterances to answer system prompts to enter product orders, telephone numbers, postal codes and addresses. Because of this difference, the interaction with the not very cooperative users revealed a problem in the speech understanding module of the Saplen system which was not observed for the other two user types: in some cases telephone numbers were correctly recognised but were understood as postal codes, while postal codes were correctly recognised but understood as telephone numbers. The reason is that the system employs its current prompt to differentiate between both kinds of utterance. Therefore when it prompts to get a telephone number, it considers that the recognised sequence of digits is a telephone number. Similarly, when it prompts for a postal code, it assumes that the sequence is a postal code, and when it prompts for a building number, it considers that the digit sequence is a building number. This simple understanding method works well if the user is very cooperative or cooperative and produces the expected kind of utterance. However the not very cooperative users may answer prompts to enter a telephone number with a product order, telephone number, postal code or address, which causes the possible confusion if the postal code is randomly selected. The problem is illustrated in the following dialogue where the postal code "18001" is initially understood as a telephone number.

(1) *Saplen*: Please say your telephone number.
(2) *User*: 1 8 0 0 1 [Recognition hypothesis: one (0.8967) eight (0.7631) zero (0.6328) zero (0.9657) one (0.8567)]
(3) *Saplen*: OK. Telephone number 1, 8, 0, 0, 1. Please say your postal code.
(4) *User*: It is wrong [Recognition hypothesis: it (0.5563) is (0.5982) wrong (0.7945)]
(5) *Saplen*: Do you want to change the telephone number? Please say yes or no.
(6) *User*: Yes [Recognition hypothesis: yes (0.8526)]
(7) *Saplen*: OK. Please say your telephone number again.
(8) *User*: 9 5 8 1 2 2 3 4 5 [Recognition hypothesis: nine (0.6025) five (0.7002) eight (0.6698) one (0.6954) two (0.9532) two (0.4235) three (0.7941) four (0.6328) five (0.8654)]
(9) *Saplen*: OK. Telephone number 9, 5, 8, 1, 2, 2, 3, 4, 5. Please say your postal code.
(10) *User*: 1 8 0 0 1 [Recognition hypothesis: one (0.7569) eight (0.6523) zero (0.5367) zero (0.8231) one (0.5314)]
(11) *Saplen*: OK. Postal code 1, 8, 0, 0, 1. Please say your address.

Dialogue 7. Problems to get the telephone number.

Because of the confusion the simulated user needed to employ turns (4)–(8) to repair the error. In principle it might not be very likely that a real user utters a telephone number when he is prompted to enter a postal code (unless he tries to make the system fail, or he is confused or nervous during the interaction). However, it is likely that when he is prompted to enter a postal code he utters a telephone number precisely to correct the misunderstanding of the telephone number that he previously uttered, as can be observed in the following interaction:

Saplen: Please say your telephone number.
User: 9 5 8 1 2 2 3 4 5
Saplen: OK. Telephone number 9, 5, 8, 1, 2, 2, 6, 3, 4, 5. Please say your postal code.
User: I said 9 5 8 1 2 2 3 4 5, not 9 5 8 1 2 2 6 3 4 5.

Analogously, it is likely that he utters a postal code to correct an error when the system is prompting to enter the address. Therefore the proposed technique has been useful to let us know that the semantic rules designed to deal with telephone numbers and postal codes must be improved to enable correct understanding even when uttered for another prompt. Because of all the problems discussed above, the average TC employing the PILM was 4.47%, which is obviously too low to consider the system performance acceptable for real users who may behave similarly to the not very cooperative users modelled in this study.

7 Limitations of the proposed technique

It is very difficult to completely and truly simulate the behaviour of a real user interacting with a dialogue system as many kinds of information and personal circumstances are involved, such as previous experience of using this technology or a priori disposition to talk to machines. Therefore the technique proposed in this paper makes the simulation in a simplified manner by means of providing responses that, depending on the type of modelled user, are more or less "adequate" in relation to the system prompts. The dialogues that can be generated are simple but useful to check very important issues to be considered in the design of spoken dialogue systems. One is whether the systems are able to recognise and understand the utterances produced by the user. Another is whether dialogue systems are able to communicate with the user via a spoken interaction as efficiently as possible, repairing possible understanding errors and providing the service required in an acceptable time. As has been shown in the previous section the proposed technique is useful for checking these issues.

However, in despite of the advantages, the technique still has several drawbacks. One is that it is only possible to simulate the behaviour of users carrying out simple and well-defined tasks. Hence its application to more sophisticated application domains which may require some kind of system-user negotiation may not be straightforward. Another limitation is that it does not take into account possible changes of mind of the user. In the current set up the simulated user tries to achieve scenario goals and is able to engage in dialogue with the Saplen system to do it. However, it has not been taken into account for example that a real user may consider the total price of the ordered products too expensive, or may not accept the estimated delivery time. For these situations a real user may try to initiate a negotiation, e.g. to remove or change some product orders. Eventually he may change his mind and abandon the conversation for several reasons. The proposed technique, in the current set up, cannot deal with these kinds of user behaviour.

Another drawback is that the implementation may not be directly portable to other application domains. Some parts of it may be re-used more or less easily (e.g. answering confirmation prompts and the management of scenario goals) but others should be implemented taking into account the specific task performed by the dialogue system. For example, at the point where the Saplen system generates a prompt to confirm the set of product orders understood and to let the proposed simulated user generate the appropriate response, we implemented a function that analyses the dialogue history and finds all the product orders understood by the system. Comparing the understood products with those in the employed scenario, the simulated user generates an affirmative or negative confirmation. This function, among others, may not be directly portable to other application types.

The proposed simulated user requires that the system developers have available a corpus of utterances related to the task to be performed by the dialogue system. Each utterance must have associated with it an orthographic transcription and a semantic representation (e.g. a frame), and it must be ensured that the corpus contains at least one utterance to answer each possible system prompt. The semantic representation must be simple enough to allow the extraction of individual data items from it to allow the answering of system prompts to get missing data and confirm data. For example, in our application domain we employed frames which allowed us to extract data items such as amount, size, taste, ingredients or temperature from product orders, as well as street name, building number or floor from addresses. As this corpus was created for previous studies we did not need to employ time to prepare it. However collecting and preparing such a corpus right from the start would be a time-consuming task which would imply an additional effort to implement the technique.

8 Conclusions and future work

In this paper we have presented a new technique to test the performance of spoken dialogue systems by artificially simulating the behaviour of three types of user (very cooperative, cooperative and not very cooperative) interacting with a system by means of spoken dialogues. Experiments have been carried out to test the performance of the Saplen system interacting with these kinds of user employing two kinds of language model for speech recognition: one based on 17 PDLMs and the other based on one PILM. Employing the technique 6,000 dialogues were automatically generated using 50 different scenarios. The evaluation results obtained from the analysis of these dialogues, set out in Table 3, show that when the PDLMs were employed the best performance was achieved for the very cooperative users. Therefore this language model should only be used for these users as otherwise the system performance would be very poor. When the PILM was employed the values for the evaluation measures were very similar for the different kinds of user. SU was very low ($< 60\%$) which caused TC to be very low too (11.67% in the best case) and thus the system performance was totally unacceptable. A second goal of the experiments was to analyse the generated dialogues to find problems in the speech recogniser, semantic analyser or dialogue manager of the system, which should be sorted out in future work to increase the system robustness. To find these problems we analysed the dialogues with very low values for the evaluation measures.

The main problem found with the speech recogniser was the difficulty it had to correctly decode some words uttered by speakers with strong southern Spanish accents. As an attempt to solve this problem we could re-train the acoustic models used by the speech recogniser with sample utterances produced by these speakers. Another possibility would be to set up a technique to replace some incorrect recognition hypotheses taking into account contextual information. This might be interesting as we observed that when the PILM was employed and the simulated user answered system confirmation prompts, the word "sí" (yes) was often substituted by the word "seis" (six), while the word "no" was often substituted by the word "dos" (two). Therefore it could be useful to implement a dialogue management strategy that, given the recognition hypothesis "seis" or "dos" with the system prompting for a "yes/no" response, makes the system generate a prompt such as "Did you say yes?". If the recognition hypothesis for this prompt was "yes" the system could assume that the simulated user generated an affirmative confirmation in the previous turn, whereas if the hypothesis was "no" the system could assume that the simulated user generated a negative confirmation. In both cases the system could proceed by carrying out the specific actions for the affirmative or negative confirmation.

The main problem found with the semantic analyser was the incorrect understanding of telephone numbers and postal codes when the system interacted with the not very cooperative users, given that correctly recognised telephone numbers were understood as postal codes and vice versa. To solve this problem it could be possible to include knowledge in the semantic rules about the different format of telephone numbers (nine digits) and postal codes (five digits). This way the semantic analyser could guess whether the utterance is a telephone number or a postal code regardless of the prompt, and the system could ask the user to confirm the guess if the utterance type does not match the current prompt.

The main problem found with the dialogue manager was the repetitive confirmation turns due to an inefficient system confirmation strategy. This happened especially when the system interacted with the cooperative and the not very cooperative users. These confirmation turns would probably not be accepted by real users as they might think the system has many problems understanding them and is very inefficient because it requires a lot of attempts to get data confirmed. To solve this problem it could be possible to implement an improved confirmation strategy that changes the prompt automatically if the system needs to repeat the confirmation turns. For example, instead of generating the prompt "I have to confirm your telephone number. Did you say 9, 5, 8, 1, 2, 2, 6, 4, 5? Please say yes or no" time after time until the user response is understood, the system could start by saying "I have to confirm your telephone number. Did you say 9, 5, 8, 1, 2, 2, 6, 4, 5?". If the response is not understood it could then generate the prompt "I have to confirm your telephone number again. Did you say 9, 5, 8, 1, 2, 2, 6, 4, 5? Please say yes or no". If the response is again not understood the system could generate the prompt "Please say again your telephone number", and if the number obtained for this last prompt matches the previously obtained number then it could assume that the telephone number has been confirmed. By implementing this range of different ways to confirm data the system would encourage the user to produce different utterance types, hence making it easier for the system to understand.

Future work to improve the proposed technique includes studying alternative methods to simulate more precisely the behaviour of real users. One possibility would be to set the type of user cooperativeness dynamically as the dialogue evolves. In the current set up this selection is made beforehand to model the behaviour of any of the three kinds of user and the setting remains fixed throughout all the dialogue. A different strategy would be to consider that a real user may change his behaviour depending on the success of the interaction. To do this the cooperativeness of the simulated user could be set to "not very cooperative" at the beginning of the dialogue and it could be changed to "cooperative" or "very cooperative" dynamically as long as the system restricts the interaction freedom as an attempt to recover from understanding problems, for example by imposing on the user a particular utterance type to confirm data (e.g. "Please say yes or no").

Other future work is concerned with enabling the simulated user's ability to model changes of mind by the user. In our application domain these changes may be related to modifications in the ordered products, which will be useful to test the system functionality that handles the product orders. Implementing this new behaviour for the simulated user will require the inclusion of utterances in the corpus expressing changes of mind as, for example, "uhm ... sorry ... instead of a large cola I'd rather prefer a small beer please". The corresponding orthographic transcriptions and semantic representations should also be included in the corpus. New scenarios should be created including as scenario goals semantic representations of this new utterance type. The implementation of the proposed simulated user should be slightly adjusted to handle this new goal type by simply including a small piece of code in part 3 of the algorithm shown in Fig. 3.

Finally, a third possibility for future work would be to generate a multimodal corpus that can be employed to enhance the user simulation. In this corpus each utterance would have associated with it the recording of the user's face when producing the utterance. A 'multimodal' simulated user could be implemented to provide both the utterance and the associated face recording to the Saplen system. This would require the conversion of the Saplen system into a multimodal system able to analyse face images and to integrate the obtained visual information with the speech recognition hypotheses. We believe that this experimental setting would be very useful to better replicate human-to-human communication, which is a multimodal process when both conversation partners speak and see each other's faces at the same time.

Acknowledgemnts The authors would like to thank the reviewers for their comments, suggestions and corrections that significantly improved the quality of this paper.

References

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

Allen J, Byron D, Dzikovska M, Ferguson G, Galescu L, Stent A (2001) Towards conversational human–computer interaction. AI Mag 22(4):27–38

Aust H, Oerder M (1995) Dialogue control in automatic inquiry systems. In: ESCA workshop on spoken dialogue systems, Vigso, Denmark, June pp 121–124

Carbini S, Delphin-Poulat L, Perron L, Viallet JE (2006) From a Wizard of Oz experiments to a real time speech and gesture multimodal interface. Signal Processing 86(12):3559–3577

Chung G (2004) Developing a flexible spoken dialog system using simulation. In Proceedings of ACL, Barcelona, Spain, pp 63–70

Cuayáhuitl H, Renals S, Lemon O, Shimodaira H (2005) Human-computer dialogue simulation using Hidden Markov models. In: Proc. of 2005 IEEE automatic speech recognition and understanding workshop (ASRU), San Juan, Puerto Rico, 28 November–1 December 2005, pp 290–295

Dow S, MacIntyre B, Lee J, Oezbek C, Bolter JD, Gandy M (2005) Wizard of Oz support throughout an interative design process. IEEE Pervasive Comput 4(8):18–26

Eckert W, Levin E, Pieraccini R (1997) User modelling for spoken dialogue system evaluation. In: Proceedings of 1997 IEEE automatic speech recognition and understanding workshop (ASRU), Santa Barbara, California, 14–17 December 1997, pp 80–87

Georgila K, Henderson J, Lemon O (2005) Learning user simulations for information state update dialogue systems. In: Proceedings of Interspeech-eurospeech, Lisbon, Portugal, 4–8 September 2005, pp 893–896

Hain T, Woodland PC, Niesler TR, Whittaker EWD (1999) The 1998 HTK system for transcription of conversational telephone speech. In: Proceedings of the 1999 international conference on acoustics, speech and signal processing, pp 57–60

Huang X, Acero A, Hon H (2001) Spoken language processing: a guide to theory, Algorithm and System Development, Prentice-Hall

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

Litman D, Pan S (2002) Designing and evaluating an adaptive spoken dialogue system. User Model User-adapt Interact 12(2/3):111–137

Lin BS, Lee LS (2001) Computer aided analysis and design for spoken dialogue systems based on quantitative simulations. IEEE Trans Speech Audio Process 9(5):534–548

López-Cózar R, Araki M (2005) Spoken, multilingual and multimodal dialogue systems. Development and Assessment, John Wiley & Sons Publishers

López-Cózar R, Callejas Z (2005) Combining language models in the input interface of a spoken dialogue system. Comput Speech Lang 20:420–440

López-Cózar R, Milone DH (2001) A new technique based on augmented language models to improve the performance of spoken dialogue systems. Proceedings of Eurospeech, pp 741–744

López-Cózar R, García P, Díaz J, Rubio AJ (1997) A voice activated dialogue system for fast-food restaurant applications. Proceedings of Eurospeech, pp 1783–1786

López-Cózar R, Rubio AJ, García P, Segura JC (1998) A spoken dialogue system based on a dialogue corpus analysis. Proceedings of LREC, pp 55–58

López-Cózar R, Rubio AJ, Díaz Verdejo JE, De la Torre A (2000) Evaluation of a dialogue system based on a generic model that combines robust speech understanding and mixed-initiative control. Proceedings of LREC, pp 743–748

López-Cózar R, De la Torre A, Segura JC, Rubio AJ, López-Soler JM (2002) A new method for testing dialogue systems based on simulations of real-world conditions. Proceedings of ICSLP, pp 305–308

López-Cózar R, De la Torre A, Segura JC, Rubio AJ, Sánchez V (2003) Assessment of dialogue systems by means of a new simulation technique. Speech Communi 40(3):387–407

McTear M (2004) Spoken dialogue technology: toward the conversational user interface. Springer, London

Möller S (2004) Quality of telephone-based spoken dialogue systems, Springer

Möller S, Englert R, Engelbrecht K, Hafner V, Jameson A, Oulasvirta A, Raake A, Reithinger N (2006) MeMo: towards automatic usability evaluation of spoken dialogue services by user error simulations. Proceedings of Interspeech, pp 1786–1789

Okamoto M, Cho K, Okamoto Y, Yamasaki T, Hattori M (2005) User-model-based adaptability evaluation for context-aware systems. In: Proceedings of the international conference of pervasive services, Santorini, Greece, 11–14 July 2005, pp 470–473

Pieraccini R, Huerta J (2005) Where do we go from here? Research and commercial spoken dialog systems. In: Proceedings of 6th SIGdial workshop on dialogue and discourse, Lisbon, Portugal, 2–3 September 2005, pp 1–10

Pietquin O, Dutoit T (2006a) A probabilistic framework for dialog simulation and optimal strategy learning. IEEE Trans Audio Speech Lang Process 14(2):589–599

Pietquin O, Dutoit T (2006b) Dynamic Bayesian networks for NLU simulation with applications to dialog optimal strategy learning. In: Proceedings of the 2006 IEEE international conference on acoustics, speech and signal processing (ICASSP), Toulouse, France, 15–19 May 2005, pp 49–52

Rabiner L, Juang BH (1993) Fundamentals of speech recognition, Prentice-Hall

Schatzmann J, Georgila K, Young S (2005a) Quantitative evaluation of user simulation techniques for spoken dialogue systems. In: Proceedings of the 6th SIGdial workshop on discourse and dialogue, Lisbon

Schatzmann J, Stuttle MN, Weilhammer K, Young S (2005b) Effects of the user model on simulation-based learning of dialogue strategies. In Proceedings of IEEE automatic speech recognition and understanding workshop (ASRU), San Juan, Puerto Rico 2005, pp 220–225

Scheffler K, Young S (2000) Probabilistic simulation of human-machine dialogues. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP), Istanbul, Turkey, pp 1217–1220

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, pp 64–70

Singh S, Litman D, Kearns M, Walker M (2002) Optimizing dialogue managment with reinforcement learning: experiments with the NJFun system. J Artif Intell Res 16:105–133

Walker M (2000) An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. J Artif Intell Res JAIR 12:387–416

Walker M, Litman D, Kamm C, Abella A (1997) PARADISE: a framework for evaluating spoken dialogue agents. In: Proceedings of the 35th annual meeting of the association for computational linguistics (ACL-97), pp 271–280

Young S (2002) Talking to machines (Statistically Speaking) In: Proceedings of ICSLP'2002, 7th international conference on spoken language processing, Denver, Colorado

Young S, Kershaw D, Odell J, Ollason D, Valtchev V, Woodland P (2000) The HTK book (for HTK Version 3.0). Microsoft Corporation