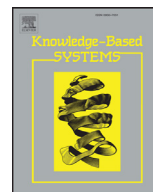




Contents lists available at ScienceDirect

## Knowledge-Based Systems

journal homepage: [www.elsevier.com/locate/knosys](http://www.elsevier.com/locate/knosys)

## Automatic creation of scenarios for evaluating spoken dialogue systems via user-simulation

Ramón López-Cózar\*

Department of Languages and Computer Systems, Faculty of Computer Science, CITIC-UGR 18071 University of Granada, Spain

## ARTICLE INFO

## Article history:

Received 2 January 2016

Revised 12 April 2016

Accepted 13 May 2016

Available online xxx

## Keywords:

Spoken dialogue systems

Speech recognition

Spoken language understanding

Natural language processing

Dialogue management

## ABSTRACT

This paper proposes a novel technique to create scenarios that can be used by a user simulator for exhaustively evaluating spoken dialogue systems. The scenarios are automatically created from simple scenario-templates that the systems' developers create manually employing their knowledge about typical goals of the system's users. The scenarios contain goals, which the user simulator will try to achieve through the interaction with the systems. The goals are represented in the form of semantic frames, which are associated with user utterances of sentences and are taken from utterance corpora. In this way, the scenarios enable speech-based interaction between the simulator and the spoken dialogue systems to be evaluated. Experiments have been carried out employing two spoken dialogue systems (Saplen and Viajero), a user simulator and two utterance corpora previously collected for two different application domains: fast-food ordering and bus travel information. Experimental results show that the technique has been useful for exhaustively evaluating the systems and finding out problems in their performance that must be addressed to improve them. Some of these problems are caused by acoustic similarity between some uttered words and strong speaker accents. Thus, we think these problems would have been difficult to uncover employing the user simulation techniques typically used nowadays, as they do not employ real speech and just consider semantics of user intentions.

© 2016 Elsevier B.V. All rights reserved.

## 1. Introduction

Spoken dialogue systems are computer programs developed to interact with users using speech to provide information or execute simple tasks [1–3]. Given that language is the most natural and efficient means of communication for humans, these systems are developed to facilitate the execution of these tasks using eyes- and hands-free devices such as microphones and telephones. These systems are typically comprised of modules to carry out five basic functions: automatic speech recognition (ASR), spoken language understanding (SLU), dialogue management (DM), natural language generation (NLG) and text-to-speech synthesis (TTS). The module that implements the ASR is called speech recogniser and its goal is to transform each sentence uttered by the user into the corresponding sequence of words. The sequence of words provided by the ASR module is the input to the SLU module. The goal of this module is to analyse the word sequence so that the system can “understand” what the user has said. To do so, many systems employ semantic rules based on word classes to obtain a conceptual representation of the sequence of words, which is typically stored

in the form of one or more semantic frames [4]. The DM module is usually called dialogue manager. It receives the frames generated by the SLU module and decides the next action to be carried out by the system, for example, provide information to the user or prompting to confirm some data (e.g., the departure city). The decision made by the dialogue manager is transformed into one or more sentences in text format by the NLG module. Finally, the TTS module transforms these sentences into the system's spoken output.

Since the evaluation and refinement of these systems with real users is typically very costly and time-consuming, a number of approaches can be found in literature focusing on user simulation techniques as alternative ways to carry out these tasks automatically [5–9]. A user simulator is a kind of dialogue system that replaces, to some extent, the behaviour of real users who interact with the dialogue system to be evaluated. Hence, this technique makes it possible to compare different system configurations, for example, in terms of dialogue management, without real users, thus saving a lot of time, financial resources and efforts of the system's designers.

In this paper we present a novel technique to create scenarios that can be used by a user simulator for exhaustively evaluating spoken dialogue systems. To the best of our knowledge, this is the

\* Corresponding author. Fax: +34 958 243179.

E-mail address: [rlopezc@ugr.es](mailto:rlopezc@ugr.es)

first paper in proposing a technique to create these scenarios. According to our technique, the scenarios are automatically created from simple scenario-templates that the systems' developers create manually employing their knowledge about typical goals of the system's users. The scenarios contain goals, which the simulator will try to achieve through the interaction with the systems. The goals are represented in the form of semantic frames, which are associated with user utterances of sentences and are taken from utterance corpora. In this way, the scenarios enable speech-based interaction between the simulator and the spoken dialogue systems to be evaluated.

### 1.1. State-of-the-art

A number of papers available in the literature focus on user simulation for the development or evaluation of spoken dialogue systems. For example, in [10] we presented a rule-based user simulator designed to interact with the Saplen spoken dialogue system, which was designed to provide fast-food information. A difference between this simulator and many others that can be found in the literature is that it provides to the dialogue system real speech (utterances of restaurant customers) instead of semantic frames. Hence, using this simulator enabled us in evaluating the dialogue system's ability in dealing with real (not simulated) errors caused by the speech recognition and spoken language understanding modules. In [11] we employed an extension of this simulator that was able to more extensively evaluating the Saplen system by simulating the behaviour of three types of user (very cooperative, cooperative, and not very cooperative). Following the idea of simulating different user types, [12] presented a user simulator that replicates the behaviour of cooperative, corrective and self-directing users.

Ai and Litman [13] employed user simulation to generate dialogues with the ITSPPOKE system, which was designed to help students to understand qualitative physics problems. In contrast to what happens with information-providing systems (e.g., for booking of travel tickets), in tutoring systems typically no clear user goals can be defined. For example, using a physics tutoring system, students usually do not have a clear goal of what physics concepts they want to learn. They may have a general goal to learn some physics, but this kind of goal is so general that it cannot help in deciding the next user action in the dialogue. Hence, the authors introduced a new model to simulate student knowledge consistency that is inspired by findings in learning research. This model constrains student performance on similar problems that require the same knowledge, i.e., concepts, principles and facts.

Ai and Weng [14] employed user simulation for generating text-based dialogues with the CHAT (Conversational Helper for Automotive Tasks) dialogue system, which supports navigation, restaurant selection and mp3 player applications. The system was specifically designed for users to interact with devices and receive services while performing other cognitive demanding, or primary tasks such as driving. The simulator kept a list of its goals and another agenda of actions to complete the goals. The application domain was restaurant selection, in which the users' tasks were to find a desired restaurant based on several constraints specified by task scenarios. The authors proposed a set of evaluation measures covering three basic aspects of task oriented dialogue systems: understanding ability, efficiency, and appropriateness of system actions.

Torres et al. [15] presented a methodology for user simulation that was applied to the evaluation and refinement of the Basurde dialogue system, which was designed to provide train information. The authors implemented a user simulator that incorporates several knowledge sources, combining statistical and heuristic information to define its dialogue strategy. The interaction between the user simulator and the dialogue system is carried out at the inten-

tion level by means of semantic frames. The simulator includes a module to keep information about the dialogue, and makes transitions in its dialogue model. In addition, it takes into account a set of rules. As in [16], the simulator can consider ASR and SLU errors by considering confidence scores to decide whether to change (or not) the slots of the frames.

Keizer et al. [17] presented a Multi-User Simulation Environment (MUSE) for automatic learning of strategies for social multi-user human-robot interaction. In this study, a robot bartender tracks multiple customers, takes their orders and serves drinks. The simulated environment replaces the vision and speech processing modules in the actual robot bartender system. It allows evaluating the behaviour of the Social Sills Executor (SSE) module, which controls the behaviour of the robot. Moreover, the simulated environment allows rapidly exploring the large space of possible actions in which the SSE module must select actions.

Ferreira and Lefèvre [18] investigated some conditions under which polarised user appraisals gathered throughout the course of a vocal interaction between a machine and a human can be integrated in a reinforcement learning-based dialogue manager. In their study they employed a user simulator in the domains of tourist information and man-robot interaction. The authors employed the agenda-based simulator from [19], which factors the user state  $S$  into an agenda  $A$  and a goal  $G$ :  $S = (A, G)$ , where  $G = (C, R)$ . Here,  $C$  denotes a set of constraints specifying the required properties that the system should satisfy, whereas  $R$  represents the desired pieces of information (e.g., post address, telephone number, available schedules). As discussed for [15] and [16], in this study the user is simulated at the intentional semantic level (i.e., dialogue act level).

The main strength of our proposal regarding these previous studies is that the scenarios that our technique creates enable real, speech-based interaction between the user simulator and the dialogue system to be evaluated. This means that the evaluation can be carried out more truthfully as it employs an utterance corpus. Thus, using our proposal system designers may find problems in system performance that perhaps might be difficult to uncover employing simulations of ASR and/or SLU errors. For example, it might not be evident to model surprising confusions between some uttered words caused by strong accents of system users or the poor quality of the speech signal. The main drawback, however, is that our technique requires the availability of an utterance corpus, which might not always be available. Moreover, collecting it typically requires a lot of time and effort for system designers.

The rest of the paper is organised as follows. Section 2 will present the proposed technique, including an examination of similarities and differences regarding previous studies available in the literature, as well as a discussion regarding the technique's application to no goal-oriented tasks. Section 3 will describe the spoken dialogue systems and the user simulator used in the experiments. Section 4 will discuss the experiments and the obtained results. Finally, Section 5 will present the conclusions and outline possibilities for future work.

## 2. The proposed technique

This paper proposes a novel technique to create scenarios that can be used by a user simulator for exhaustively evaluating spoken dialogue systems. The scenarios are automatically created from simple scenario-templates that the systems' developers create manually using their knowledge about typical goals of the system's users. The scenarios contain goals, which the user simulator will try to achieve through the interaction with the systems. It should be noted that the technique does not find system problems directly. Its objective is to provide a means for automatically

generating massive amounts of dialogues employing scenarios. After the generation, systems' developers must analyse the dialogues to find out the problems. For example, using the parameters that will be discussed in Section 2.1.1, the developers can specify the generation of dialogues to be cancelled if the systems do not work well. Then, it is the developers' task to analyse the dialogues and find out the reasons for the incorrect system performance.

To create the scenarios we consider three sets to model the knowledge of the systems' developers about the activity of the users. First, a set of goal types  $G_T = \{gt_1, gt_2, gt_3, \dots\}$  where each goal type represents a general intention of the users of a dialogue system. For example, if the system is designed to provide bus travel information, some user intentions can be the following: provide telephone number ( $gt_1 = \text{provideTelephoneNumber}$ ), provide destination city ( $gt_2 = \text{provideDestinationCity}$ ), get information about travel duration ( $gt_3 = \text{getTravelDuration}$ ), get information on bus departures ( $gt_4 = \text{getBusDepartures}$ ).

Secondly, we consider a set of goals  $G = \{g_1, g_2, g_3, \dots\}$  where each goal represents a specific intention of the users of the dialogue system, for example, the goal  $g_1$  can be:  $g_1 = \text{"getting bus departure schedules from Madrid to Barcelona on direct buses"}$ . The proposed technique considers that each goal is represented in the format of a semantic frame. In this format, the goal  $g_1$  can be represented as follows:

```
<action> = "departureSchedulesQuery"
<departureCity> = "Madrid"
<destinationCity> = "Barcelona"
<busType> = "direct"
```

The technique will create scenarios containing this kind of goals. The goals will be associated with users' real utterances, which will be used by the user simulator in its interaction with the dialogue system to be evaluated.

Thirdly, we consider a set of utterance types  $U_T = \{ut_1, ut_2, ut_3, \dots\}$ , where each type represents the semantics of user utterances. For example, we can consider that the type of the utterance "to Barcelona" is  $ut_2 = \text{destinationCity}$ , the semantics of the utterance "please one ham sandwich" is  $ut_4 = \text{foodOrder}$ , and the semantics of the utterance "Elm street number 13" is  $ut_5 = \text{postAddress}$ .

In addition to the sets regarding the activity of the users, we consider a set to model the knowledge of the systems' developers about the activity of the dialogue system to be evaluated. This is a set of prompt types  $P_T = \{pt_1, pt_2, pt_3, \dots\}$  where each type represents the type of data that the system requests from the user. For example, we can consider that the type of the prompt "Please say your telephone number" is  $pt_1 = \text{requestTelephoneNumber}$  and the type of the prompt "Did you say your postcode is 18001? Please say yes or no" is  $pt_2 = \text{confirmPostcode}$ .

## 2.1. The scenario-template

To create the scenarios that will be used for evaluating spoken dialogue systems, the proposed technique uses a scenario-template that, generally speaking, can be as shown in Fig. 1. As can be observed, the template is comprised of four sections to deal with parameters for dialogue generation cancellation, in-domain goals, out-of-domain goals, and required goals. The first section contains data that will be included "as-is" in the scenarios that the technique will create. The other sections contain patterns that will enable to include in the scenarios different goals. One advantage of this scenario-template is that it allows creating scenarios for any application domain. Another advantage is that the scenario-template does not include any goals regarding confirmations and error indications to be made by the user simulator during its interaction with the dialogue system to be evaluated. This is so because users of dialogue systems typically carry out both types of action

when needed regardless of the systems' application domains. As these actions can be considered scenario-independent goals, our user simulator can confirm data and initiate error recovery strategies automatically when needed. For example, if the dialogue system generates the confirmation prompt: "Did you say you want to travel to Barcelona?", the simulator will answer "Yes" or "No" depending on the scenario goals. Also, if the simulator detects a system misunderstanding of a goal, in its next dialogue turn it may either provide the missing data or warn the system about the error by generating a prompt such as: "You made a mistake". In the following we will explain the four sections of the scenario-template. Fig. 6 and Fig. 7 show two sample scenario-templates similar to the ones that we have used in our experiments.

### 2.1.1. Parameters for dialogue generation cancellation

Section 1 of the scenario-template contains the parameters for dialogue generation cancellation that will be included in the scenarios that the proposed technique will create. We use these parameters to consider that real users of dialogue system will likely hang up the telephone if the systems are not working well. In this way, if during the interaction between the user simulator and a dialogue system the value of any of these parameters exceeds a given limit (represented as value in the scenario-template), the simulator will cancel the current dialogue and include a tag in the log file that will store the dialogue, reporting that the interaction was not successful. As can be observed in Fig. 1, the parameters for dialogue generation cancellation are as follows:

- **numMaxDialogueTurns:** This is the maximum number of dialogue turns, including both dialogue system's turns and user simulator's turns.
- **numMaxPromptsToGetDataTypes:** This is the maximum number of dialogue system's prompts (i.e., system attempts) to get from the user simulator a given data type. For example, the system can make up to three attempts to get a telephone number from the simulator.
- **numMaxPromptsToConfirmDataTypes:** This is the maximum number of dialogue system's prompts (i.e., system attempts) to get confirmed a given data type by the user simulator. For example, the system can make up to two attempts to get a confirmation for the telephone number already provided by the simulator.

### 2.1.2. In-domain and out-of-domain goals

Sections 2 and 3 of the scenario-template allow including in the scenarios that the technique will create what we call *in-domain* and *out-of-domain* goals, which requires having available an utterance corpus representative for each domain. It should be noted that it is not mandatory to include out-of-domain goals in the scenario-template. However, doing so makes it possible evaluating more exhaustively the dialogue system, as will be explained next.

The in-domain goals are goals regarding the application domain for which the dialogue system to be evaluated has been designed. These goals makes it possible to simulate, to some extent, the behaviour of cooperative users who answer system prompts as the system expects, for example, uttering a telephone number when the system prompts them to do so. As can be observed in Fig. 1, the scenario-template specifies the utterance corpora from which the goal types must be taken, as well as the utterance types associated with the goal types.

The out-of-domain goals are goals regarding application domains for which the dialogue system to be evaluated has not been designed. These goals makes it possible to model, to some extent, the behaviour of non-cooperative or inexperienced users who might not answer system prompts as the system expects. As for the in-domain goals, the scenario-template specifies the utterance corpora from which the out-of-domain goal types must be

# Section 1: Parameters for dialogue generation cancellation				
numMaxDialogueTurns = value <sub>1</sub>				
numMaxPromptsToGetDataTypes = value <sub>2</sub>				
numMaxPromptsToConfirmDataTypes = value <sub>3</sub>				
# Section 2: In-domain goals				
utteranceCorpus = name <sub>1</sub>	goalType = gType <sub>1a</sub>	utteranceTypes = uTypes <sub>1a</sub>		
utteranceCorpus = name <sub>1</sub>	goalType = gType <sub>1b</sub>	utteranceTypes = uTypes <sub>1b</sub>		
utteranceCorpus = name <sub>1</sub>	goalType = gType <sub>1c</sub>	utteranceTypes = uTypes <sub>1c</sub>		
...				
# Section 3: Out-of-domain goals				
utteranceCorpus = name <sub>2</sub>	goalType = gType <sub>2a</sub>	utteranceTypes = uTypes <sub>2a</sub>	promptType = pType <sub>2a</sub>	probOODResp = prob <sub>2a</sub>
utteranceCorpus = name <sub>2</sub>	goalType = gType <sub>2b</sub>	utteranceTypes = uTypes <sub>2b</sub>	promptType = pType <sub>2b</sub>	probOODResp = prob <sub>2b</sub>
utteranceCorpus = name <sub>2</sub>	goalType = gType <sub>2c</sub>	utteranceTypes = uTypes <sub>2c</sub>	promptType = pType <sub>2c</sub>	probOODResp = prob <sub>2c</sub>
...				
utteranceCorpus = name <sub>3</sub>	goalType = gType <sub>3a</sub>	utteranceTypes = uTypes <sub>3a</sub>	promptType = pType <sub>3a</sub>	probOODResp = prob <sub>3a</sub>
utteranceCorpus = name <sub>3</sub>	goalType = gType <sub>3b</sub>	utteranceTypes = uTypes <sub>3b</sub>	promptType = pType <sub>3b</sub>	probOODResp = prob <sub>3b</sub>
...				
utteranceCorpus = name <sub>4</sub>	goalType = gType <sub>4a</sub>	utteranceTypes = uTypes <sub>4a</sub>	promptType = pType <sub>4a</sub>	probOODResp = prob <sub>4a</sub>
...				
# Section 4: Required goals				
utteranceCorpus = name <sub>1</sub>	goalType = gType <sub>1d</sub>	utteranceTypes = uTypes <sub>1d</sub>		
utteranceCorpus = name <sub>1</sub>	goalType = gType <sub>1e</sub>	utteranceTypes = uTypes <sub>1e</sub>		
utteranceCorpus = name <sub>1</sub>	goalType = gType <sub>1f</sub>	utteranceTypes = uTypes <sub>1f</sub>		
...				

Fig. 1. Structure of the scenario-template to create scenarios for evaluating spoken dialogue systems.

taken, as well as the utterance types associated with the each goal type. To enable an ample range of user simulator's behaviours, the scenario-template uses the **probOODResp** parameter to assign a probability of out-of-domain response to each out-of-domain goal. Depending of the value for this parameter, the user simulator will employ either the out-of-domain goal or the in-domain goal to answer the corresponding dialogue system's prompt. For example, we can specify that it is very likely that for a dialogue system's prompt to enter fast-food orders (e.g., "Would you like to eat something?"), the user simulator will answer by making a query about bus departures. The value 1.0 means that the user simulator will always provide an out-of-domain response for the system's prompt type, which represents the interaction of a very little cooperative user; for example, the simulator will always provide a food order when the system prompts to enter a telephone number. In contrast, a value of 0.0 means that the simulator will never provide an out-of-domain response for the prompt type, which represents the interaction of a very cooperative user; for example, the simulator will always provide a telephone number when the dialogue system prompts to do so.

Another difference between these two types of goal is that our user simulator will insist in making the dialogue system understand correctly the in-domain goals. This means that the simulator will warn about system misunderstandings and will answer system's confirmation prompts. However, the simulator will not insist in making the dialogue system understand the out-of-domain goals; it will just "utter" the out-of-domain sentences to make the system deal with not expected sentence types and vocabulary. Thus, out-of-domain goals can be representative of user uncertainty about what to say next or user attempts to deliberately check the robustness of the system. Therefore, these goals can be useful for evaluating whether the system is able to properly recover from this type of input. Sample in-domain and out-of-

domain goals can be observed in Sections 2 and 3, respectively, of Fig. 6 and Fig. 7.

It should be noted that if the systems' developers decide to include out-of-domain goals in the scenario-template, they must pay attention the possible overlapping between the spoken corpora that will be used. The proposed technique will work exactly the same with very similar and very different corpora, i.e., the scenarios will be created exactly in the same way. However, there would be a difference in the evaluation results. For example, let us consider that the developers employ for the in-domain goals a corpus about air travel information, and for the out-of-domain goals a corpus about bus travel information. Hence, given the similarities between the vocabulary and the grammar of the sentences, it would be likely that a system to be evaluated understands correctly the goal expressed by the sentence "I want to travel from Boston to New York" taken from the out-of-domain corpus. On the opposite, employing a clearly different corpus for the out-of-domain goals (e.g., one about fast-food ordering) would make the system to deal with not expected vocabulary and grammars, thus leading to worse evaluation results but better evaluation of the system's robustness in recovering from unexpected user input.

### 2.1.3. Required goals

Finally, Section 4 of the scenario-template allows including what we call *required* goals in the scenarios that the proposed technique will create. To include these goals we just need the corpus employed for the in-domain goals. The objective of the required goals is to account for possible missing data in the utterances taken from the corpus. Hence, these goals will ensure that the scenarios will contain all the data the user simulator will need to accomplish a given task during the interaction with the dialogue system. For example, let us consider that a task for the



user simulator is to book a bus travel from one city to another, which obviously will need that the simulator provides to the system the departure and the destination cities. However, it might be possible that some utterances making these bookings taken from the corpus just contain one of the two cities, for example, “Please I would like to book two seats for Madrid on Thursday on a direct bus”. If this utterance was used as an in-domain goal, there would not be any data in the scenario regarding the departure city, in which case the user simulator would not be able to answer the system prompt “Please say the departure city”. To prevent this from happening, the system designers who want to evaluate the system must specify in the scenario-template that the departure city is a required data for carrying out the bus travel booking task. Doing so, this data will be automatically included in the scenario, if necessary, as another goal.

Sample required goals are shown in Section 4 of Fig. 6 and Fig. 7. As can be observed, these goals complement the in-domain goals so that the scenarios contain all the necessary data for carrying out the tasks specified by the in-domain goals. For example, Fig. 6 shows in-domain goals to make food order and drink order. Since not all the food or drink orders in our utterance corpus specify the amount of product, we include the amount of product as a required goal in the scenario-template. In this way, if the utterance finally selected for the drink order goal contains the amount data (e.g., “one Fanta”), the proposed technique will not include this data in this scenario, as it will be already available in the in-domain goal. On the opposite, if the selected utterance does not contain the amount data (e.g., “beer please”), our technique will include it in the scenario as an additional goal. Doing so we ensure that the scenario will contain the amount data regardless of the utterance finally used for the drink order goal.

## 2.2. Implementation of the proposed technique

The proposed technique can be implemented following three steps: initialisation of the so-called *corpora model*, processing of the corpora model, and creation of scenarios, as will be described in the following sections. In addition to the scenario-template discussed in the previous section, the implementation requires the definition of the following parameters: folder to store the scenarios that the proposed technique will create, name of the spoken dialogue system to be evaluated using the scenarios, and paths to the corpora containing the utterances that the user simulator will use for interacting with the system. The implementation considers that for each utterance of a sentence there are three files in the corpus that only differ in the file extension. For example, the files associated with an utterance of a telephone number in our Saplen corpus are the following: `telephoneNumber_20040312_19-52-39.wav` (speech samples), `telephoneNumber_20040312_19-52-39.txt` (orthographic transcription), and `telephoneNumber_20040312_19-52-39.RScor` (semantic frame)

### 2.2.1. Initialisation of the so-called corpora model

Fig. 2 shows an algorithm<sup>1</sup> to carry out this step. It takes as input the scenario-template and the name of the dialogue system to be evaluated, and produces as output what we call a *corpora model*. This model provides information about the utterance corpora and the dialogue system's prompts that will be used for creating the

scenarios. First, the algorithm initialises the corpora model creating a blank list of records that contain fields to store the following data:

- **Dialogue system.** This field contains the name of the dialogue system to be evaluated, for example, Saplen.
- **Utterance corpus.** This field stores the name of the corpus that contains the utterances of type `utteranceType` which will be used by the user simulator for interacting with the dialogue system. For example, this name can be `saplen` and the utterance type can be `telephoneNumber`.
- **Goal type.** This field stores a goal type  $gt_i \in G_T$  (see Section 2), for example, `provideTelephoneNumber`.
- **Required.** This field specifies whether the goal type is in the “Required goals” section of the scenario-template, as explained in Section 2.1.3.
- **Utterance types.** This field stores the utterance types  $ut_i \in U_T$  (see Section 2) that the dialogue system's developers associate with a goal type  $gt_k$  by means of a function that we call  $\Omega$ . For example,  $\Omega(gt_1) = \{ut_1\}$ ,  $\Omega(gt_2) = \{ut_2, ut_3\}$ , where  $gt_1 = \text{provideTelephoneNumber}$ ,  $gt_2 = \text{provideDestinationCity}$ ,  $ut_1 = \text{provideTelephoneNumber}$ ,  $ut_2 = \text{provideDestinationCity}$ ,  $ut_3 = \text{provideCity}$ .
- **Prompt type.** This field is only used for out-of-domain goals and determines the dialogue system's prompt type  $pt_i \in P_T$  for which the user simulator will provide an out-of-domain response, as discussed in Section 2.1.2.
- **Probability for out-of-domain response.** This field is only used for out-of-domain goals and specifies how likely the user simulator will generate an out-of-domain response, as discussed in Section 2.1.2.
- **Counter of semantic frames.** This field's data is not extracted from the scenario-template and is used to let us know how many semantic frames for the specified utterance type there are available in the utterance corpus. For example, the corpus may have 500 frames for the utterance type  $ut_1 = \text{telephoneNumber}$ . It is worth noting that some of these frames may be identical if the recorded speakers uttered the same telephone number.
- **Set of semantic frames.** This field stores a set that contains all the files available in the corpus containing semantic frames for the specified utterance type, for example, telephone numbers.

Secondly, the algorithm reads the scenario-template and takes from it the values for the parameters for dialogue generation cancellation (Section 1 of the template) as well as the data regarding the in-domain, out-of-domain and required goals. Finally, the algorithm stores all this data in the fields of the corpora model.

### 2.2.2. Processing of the corpora model

Fig. 3 shows an algorithm to carry out the processing of the corpora model. It receives the initialised corpora model, the utterance corpora, and the name of the dialogue system to be tested. As a result, it decides the largest set of utterance types and provides a modified version of the input corpora model.

To process the corpora model, the algorithm first takes each utterance type in the model and, if this type is used for in-domain goals then it creates a set that stores all the semantic frames of the type, for example, set of frames associated with telephone numbers. Second, the algorithm finds out which is the largest set of utterance types and stores its size if the output variable `largestUtteranceTypesSet`.

### 2.2.3. Creation of scenarios

This step takes the corpora model provided by the previous step and creates the scenarios. As discussed in Section 2, the scenarios contain the goals that the user simulator will try to achieve during

<sup>1</sup> In this algorithm, `NUM_MAX_UTTERANCE_TYPES` is a constant value to enable enough storage space to accommodate all the utterance types available in the utterance corpora that will be used for evaluating the dialogue system, for example, telephone numbers, city names, post addresses, food names, street names, etc. In our experiments, this parameter is set to 100 to allocate more than enough room.

**Algorithm** initialiseCorporaModel ( **Input:** scenarioTemplate, dialogueSystemToBeEvaluated; **Output:** corporaModel )  
**Begin**

/\* Section 1: Creation of blank corpora model \*/

```
for j = 1 to NUM_MAX_UTTERANCE_TYPES
  corporaModel[ j ].dialogueSystem = " "
  corporaModel[ j ].utteranceCorpus = " "
  corporaModel[ j ].goalType = " "
  corporaModel[ j ].required = " "
  corporaModel[ j ].utteranceTypes = " "
  corporaModel[ j ].promptType = " "
  corporaModel[ j ].probOODResp = " "
  corporaModel[ j ].counterSemanticFrames = 0
  corporaModel[ j ].setOfSemanticFrames = " "
endfor
```

/\* Section 2: Reading of parameters for dialogue generation cancellation \*/

```
read ( scenarioTemplate, numMaxDialogueTurns )
read ( scenarioTemplate, numMaxPromptsToGetDataTypes )
read ( scenarioTemplate, numMaxPromptsToConfirmDataTypes )
```

/\* Section 3: Reading of in-domain, out-of-domain and required goals, as well as writing data in the corpora model \*/

```
while not end ( scenarioTemplate )
  read ( scenarioTemplate, utteranceCorpus, goalType, required, utteranceTypes, promptType, probOODResp )
  if ( inRequiredGoals ) then required = "Yes" else required = "No" endif
  if ( promptType == " " ) then dialogueSystem = dialogueSystemToBeEvaluated else dialogueSystem = " " endif
  write ( corporaModel, dialogueSystem, utteranceCorpus, goalType, required, utteranceTypes, promptType, probOODResp )
endwhile
End
```

**Fig. 2.** Algorithm to initialise the corpora model.

**Algorithm** processCorporaModel ( **Input:** corporaModel, utteranceCorpora, dialogueSystemToBeEvaluated;  
**Output:** largestUtteranceTypesSet, corporaModel )

**Begin**

```
for j = 1 to NUM_MAX_UTTERANCE_TYPES
  dialogueSystem = corporaModel[ j ].dialogueSystem
  utteranceCorpus = corporaModel[ j ].utteranceCorpus
  utteranceTypes = corporaModel[ j ].utteranceTypes
```

/\* Section 1: Creation of sets of semantic frames that will be used as in-domain goals \*/

```
for each ( file.wav ∈ utteranceTypes ) in utteranceCorpus
  if ( dialogueSystem == dialogueSystemToBeEvaluated ) then
    insert ( corporaModel[ j ].setOfSemanticFrames, file.RScor )
    corporaModel[ j ].counterSemanticFrames = corporaModel[ j ].counterSemanticFrames + 1
  endif
endfor
endfor
```

/\* Section 2: Decision on largest utteranceTypes set \*/

```
largestUtteranceTypesSet = 0
for j = 1 to NUM_MAX_UTTERANCE_TYPES
  size = corporaModel[ j ].counterSemanticFrames
  if ( size > largestUtteranceTypesSet ) then largestUtteranceTypesSet = size endif
endfor
End
```

**Fig. 3.** Algorithm to process the corpora model.

```

Algorithm createScenarios ( Input: largestUtteranceTypesSet, dialogueSystemToBeEvaluated, numMaxDialogueTurns,
numMaxPromptsToGetDataTypes, numMaxPromptsToConfirmDataTypes; Output: setsOfScenarios )
Begin
  for i = 1 to largestUtteranceTypesSet
    new ( scenario )

    /* Section 1: Inclusion of parameters for dialogue generation cancellation */

    write ( scenario, "# Section 1: Parameters for dialogue generation cancellation" )
    write ( scenario, "numMaxDialogueTurns=", numMaxDialogueTurns )
    write ( scenario, "numMaxPromptsToGetDataTypes=", numMaxPromptsToGetDataTypes )
    write ( scenario, "numMaxPromptsToConfirmDataTypes=", numMaxPromptsToConfirmDataTypes )

    /* Section 2: Inclusion of in-domain and out-of-domain goals */

    for j = 1 to NUM_MAX_UTTERANCE_TYPE
      getDataFromCorporaModel ( j, dialogueSystem, utteranceCorpus, goalType, required, utteranceTypes, promptType, probOODResp,
        setOfSemanticFrames )
      if ( required == "No" ) then
        F = searchForLeastUsedSemanticFrame ( setOfSemanticFrames, goalType )
        if ( dialogueSystem == dialogueSystemToBeEvaluated ) then
          if ( not section2included ) then write ( scenario, "# Section 2: In-domain goals" ) endif
          write ( scenario, "# ", F.Path, "UseCount = ", F.useCount, "utteranceCorpus = ", utteranceCorpus, " goalType = ", goalType )
        else
          if ( not section3included ) then write ( scenario, "# Section 3: Out-of-domain goals" ) endif
          write ( scenario, "# ", F.Path, "UseCount = ", F.useCount, "utteranceCorpus = ", utteranceCorpus, " goalType = ", goalType, " systemPrompt = ",
            systemPrompt, " probOODResp = ", probOODResp )
        endif
        write ( scenario, F.frame )
      endif
    endfor

    /* Section 3: Inclusion of required goals */

    for j = 1 to NUM_MAX_UTTERANCE_TYPE
      getDataFromCorporaModel ( j, dialogueSystem, utteranceCorpus, goalType, required, utteranceTypes, promptType, setOfSemanticFrames )
      if ( ( required == "Yes" ) and ( not FoundInScenario ( scenario, goalType ) ) ) then
        F = searchForLeastUsedSemanticFrame ( setOfSemanticFrames, goalType )
        if ( not section4included ) then write ( scenario, "# Section 4: Required goals" ) endif
        write ( scenario, "# ", F.Path, "UseCount = ", F.useCount, "utteranceCorpus = ", utteranceCorpus, " goalType = ", goalType )
        write ( scenario, F.frame )
      endif
    endfor
  endfor
End

```

Fig. 4. Algorithm to create scenarios.

its interaction with the dialogue system to be evaluated. The goals are semantic frames associated with user utterances in the corpora. To ensure that all the frames in the corpora used for in-domain goals will be used, the number of scenarios that will be created is determined by the utterance type in the in-domain corpus that contains the largest number of semantic frames. For example, if there are 20,000 frames for telephone numbers and this is the largest utterance type, this step will create 20,000 scenarios, one containing each frame. This implies that frames of other utterance types may be used several times. To do so, this step keeps trace of the number of times each frame has been included in the scenarios. Fig. 4 shows an algorithm to implement this step. It takes as input the information regarding the largest utterance type, the name of the dialogue system to be evaluated, and the values for the parameters for dialogue generation cancellation, and produces as output the scenarios. Fig. 8 and Fig. 9 show two sample scenarios that have been used to evaluate our experimental Saplen and Viajero dialogue systems, respectively.

As can be observed in Fig. 4, the algorithm employs the first for...endfor loop to create the number of scenarios necessary to use at least once each semantic frame in the corpus used for in-domain goals. Within this loop, the algorithm first creates a new scenario and writes in it the parameters for dialogue generation cancellation.

The algorithm's Section 2 gets data from the created corpora model regarding each goal type in the scenario-template. If the goal type is not in the template's section for required goals (i.e., Section 4 of Fig. 1), the algorithm search for the least used semantic frame of the corresponding goal type (e.g., least used frame of a

telephone number) and writes in the scenario the path of the file containing the frame as well as the number of times it has been used. Next, the algorithm checks whether the goal is either in-domain or out-of-domain. In the first case, it writes in the scenario the data regarding the utterance corpus from which the frame has been extracted (e.g., Saplen) and the goal type (e.g., makeFoodPriceQuery). In the case of out-of-domain goal, the algorithm writes the same data as in the previous case, plus the data regarding the dialogue system's prompt type (e.g., requestPostcode) for which the out-of-domain goal will be used, as well as the probability for generating the out-of-domain response. The third section of the algorithm is similar as Section 2, the only difference being in that it focuses on required goals only.

### 2.3. Differences and similarities regarding previous studies

To the best of our knowledge, this is the first paper in proposing a technique to create scenarios for evaluating spoken dialogue systems. Thus, it has not been possible for us to focus this section on differences and similarities between techniques for creating the scenarios, which is the novelty of the paper. Therefore, in this section we focus on similarities and differences between our study and others in terms of the approach to user simulation and the scenarios employed for the simulation.

A number of papers on user simulation model the dialogue at the intention level, which means that at each dialogue state the simulator provides to the dialogue system some sort of semantic frame for each user intention. This approach has the advantage of avoiding the need for employing human language in the form of

speech signals or word sequences. Typically, the type of frame that the simulator provides is decided from a statistical training, which is carried out employing a dialogue corpus. For example, this is the approach followed by the bigram model [20], Levin model [21], Scheffler model [22] and Pietquin model [23]. Opposite to these approaches, we propose to use real user speech in the interaction between user simulators and spoken dialogue systems. In this way the systems can be exhaustively evaluated facing real (not simulated) problems concerned with ASR, and with the effects that these problems provoke in SLU and DM.

Our proposal is related to a number of papers available in the literature in which user simulators employ scenarios to interact with spoken dialogue systems. For example, [15] defined 14 scenarios concerned with queries about timetables, prices and/or train types of one-way trips. Using these scenarios and varying some parameters in the experimentation (including or not ASR and/or understanding errors) the authors ran several series of a few thousand simulated dialogues. A similarity between their work and ours relies in that each scenario is represented as a set of attribute-value pairs, for example, `destinationCity = "Barcelona"`. [16] used a set of scenarios to generate simulated dialogues with the rule-based DM modules of four dialogue systems (DI@L-log, UAH, LUNA and EDECAN). These dialogues were used to learn the dialogue models of the systems and train statistical DMs. A similarity between these two works and ours is that the scenarios contain goals of real users taken from dialogue corpora for the systems. A difference is that in the other studies the scenarios are hand-crafted whereas in our work they are automatically created using utterance corpora.

Our work is related to [24] on the student tutoring application domain, in which the authors employ a "Cluster Model" and a "ProbCorrect Model". The former is used to generate a student answer based on both the content of the tutor's question and the student's previous answers to a similar question. This model assumes that a student will have a higher chance to give a correct answer to the question of a cluster in which the student mostly answers correctly before, and a lower chance to do so otherwise. The "ProbCorrect Model" is designed to give a correct/incorrect with the same probability as the average of the real students, which is similar to the bigram model described in [21]. A similarity between their work and ours is that both employ a probability to decide the answer that the user simulator will provide to the dialogue system. In our work, depending of the value for this probability (`probOODResp` parameter used in the scenario-template, see Fig. 1), the answer will be either in-domain or out-of-domain, which can be considered analogous to the "correct/incorrect" answer in [24].

Our study also has similarities with the study by [14], in which the authors experimented with the CHAT text-based dialogue system to find desired restaurants based on several constraints specified by task scenarios. In their study, as in ours, the user simulator tries to achieve goals. A difference is that in theirs, at the beginning of each dialogue the simulator randomly generates an agenda for the list of ordered goals corresponding to the constraints in requesting a restaurant, whereas in our study the user simulator does not generate any goals. Instead, it uses the scenarios created by the proposed technique, which contain the goals.

Our proposal has also similarities with [23] and [8] since like in our proposal, in these two works goals are represented as attribute-value pairs, for example, `destinationCity = "Barcelona"`. A difference is that in our proposal goals are created from the analysis of utterance corpora, whereas in these studies the values for the attribute-value pair are randomly selected from a database. Moreover, [23] attaches a simple counter to each attribute-value pair to record how often a piece of information has been transmitted to the system, which is used to model the likelihood of

the user hanging up the telephone before completing the task. The objective of this counter is similar to that of the parameters `numMaxPromptsToGetDataTypes` and `numMaxPromptsToConfirmDataTypes` that we use for dialogue generation cancellation in our scenario-template (see Section 1 of Fig. 1).

Our work has also similarities with [17] since both employ a method to model that a user may give up the conversation. Concretely, we use the parameters mentioned above whereas [17] employ a "patience level" that is reduced in every frame that the user is bidding for attention or being served by the system. A difference is that in their study each user simulator is initialised with a random goal (a type of drink they want to order) whereas in ours the goals are not created at random but from the analysis of utterance corpora. Also, the scenarios that our technique creates contain several goals instead of just one. Note that they would contain just one goal if the scenario-template just contained one goal. Hence, we could say that our technique is more flexible.

#### 2.4. Application of the technique to no goal-oriented tasks

The proposed technique is suited for evaluating spoken dialogue systems developed for goal-oriented tasks. This is why the scenarios that it creates represent typical goals of users, such as getting specific information or carry out a determined task. A typical evaluation measure for these systems is task completion (TC), which we have used in our experiments. On the contrary, in no goal-oriented tasks the users do not have any specific goal in mind when they interact with the systems. Moreover, the systems are not developed to provide a specific information or service but, for example, to establish a social relationship with the user. Another evaluation measures are employed for evaluating these tasks, for example, to what extent the systems are able to change their dialogue strategies to preserve user engagement and allow the conversation to continue.

Thus, if we wanted to apply our technique to no goal-oriented tasks, the proposed scenario-templates should not be concerned with generating user goals, but for example, user's emotional states, wishes to keep on (or stop) interacting with the systems, and desires to talk about another topics. Moreover, it should be necessary to develop alternative models to take into account interaction factors and/or personal characteristics of the users, which should be represented some how in the templates. The scenarios that the technique would create should represent users who may change their emotional state and interest in the current conversation throughout the dialogue. In this way, the dialogue systems could be evaluated considering to what extent they are able to address the users' emotional states and change the conversation topics, if necessary, so that the users keep interested in the conversation.

### 3. Experimental spoken dialogue systems and user simulator

This section describes briefly the two spoken dialogue systems previously developed in our lab (Saplen and Viajero) as well as the user simulator that we have used in the experiments for evaluating both systems employing the proposed technique. The goal of the Saplen system is to answer telephone-based orders and queries made by clients of fast-food restaurants, whereas that of Viajero is to provide bus travel information and register travel bookings.

The dialogue management for the Saplen system is as follows. After an initial welcome message, the system prompts the user to enter product orders or queries. When products have been ordered the system confirms the orders, informs of the total price and prompts the user to accept it. Then 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



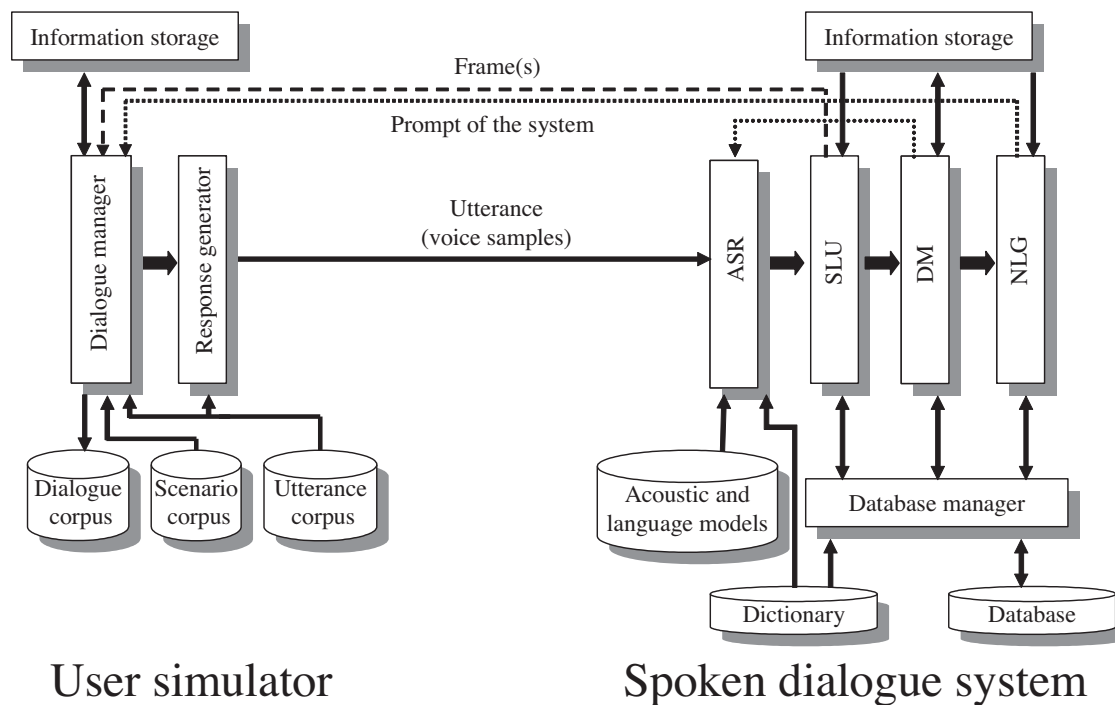


Fig. 5. User simulator interacting with a spoken dialogue system.

of known clients. If it is, it then prompts the user to confirm the stored data; otherwise it prompts for the user's postcode and confirms it. The system then prompts for the user's post address and confirms the data obtained. If the user confirms them, they are stored in the 'known user' 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, street name, building number, building floor and apartment letter.<sup>2</sup> At the end of the dialogue the system gives the estimated delivery time and concludes by thanking the user for the call.

For the Viajero system the dialogue management is as follows. After an initial welcome message the system prompts whether the user wants either information, make a travel reservation or other type of service. If the user selects information, the system prompts to select the type of query. The user may ask for arrival and departure schedules, travel prices and duration, etc. If the user wishes to make a booking the system prompts for their telephone number and confirms it. Then it prompts for the required data for the booking (departure and arrival city, departure date and time, type of bus and number of seats to reserve). Next the system prompts the user to confirm all the data in the booking. If the confirmation is affirmative the system registers the booking in a database and provides a booking code with some additional information for the user. If the confirmation is negative the system asks the user to identify the incorrect data and starts a correction sub dialogue. When the task being done is achieved (e.g., travel booking) the system prompts whether the user wants anything else. If the response is affirmative the system prompts for the new service wished by the user, whereas if it is negative the system thanks the user for the call and farewells.

<sup>2</sup> These are the data typically used in Spain for the post 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"), street name (e.g., "Elm") and building number (e.g., "13").

Using both systems the user is informed at the beginning of the dialogue that to correct system misunderstandings they must enter error-indication sentences such as "error", "mistake", "it is incorrect" or "you made a mistake". This kind of sentence makes the systems remove the data obtained in the previous interaction turn and return to the previous dialogue state. Both systems employ mixed-initiative interaction for the dialogue, which means that both the system and the user can take the initiative in the dialogue at any time. In this way, when the Saplen system prompts the user for their telephone number, they can provide the number or other types of sentence, for example, a product order. The order causes a focus shift in the dialogue which makes the system try to get all the data required for a complete order. When the data is obtained, the system goes back to the previous focus in the dialogue and prompts again for the telephone number. The user responses for both systems can be over informative, which means that the user can provide in one interaction more data than required by the system. For example, when the Viajero system prompts for the destination city in a booking, the user may provide the city name and the departure time.

### 3.1. User simulator

As was discussed in Section 1, evaluating spoken dialogue systems with real users is a very costly and time-consuming task. To address this problem the scientific community started some years ago to use a simulation technique in which another system, called *user simulator*, mimics to some extent the behaviours of real users in the interaction with the systems. Hence, this technique allows generating large amounts of evaluation data in low-cost and time-efficient manner [14,24–26]. In the experiments presented in this paper, our two spoken dialogue systems interact with the user simulator that we developed in a previous work [10] and have used in a previous studies [11,27,28]. Fig. 5 shows the interaction between the user simulator and a spoken dialogue system.

The simulator automatically interacts with the dialogue system and generates dialogues that are stored in a dialogue corpus for

**Table 1**  
Utterances used for training and evaluating the Saplen system.

Sentence type	Utterances	Time (min)	Size (MB)
Food order	810	45	50.1
Drink order	617	40	44.2
Price query	608	45	48.3
Ingredients query	416	34	36.3
Post address	4576	441	463.5
Street name	4647	242	263.5
Post address type	543	25	28.1
Building number	3478	173	194.2
Apartment letter	906	49	55.2
Building floor	333	18	20.3
Information needed	447	26	29.3
Product temperature	260	22	29.7
Postcode	501	30	33.4
Yes/No confirmation	5346	157	186.8
Product amount	610	22	26.7
Food name	285	13	14.7
Food ingredients	508	24	27.7
Drink name	360	16	18.1
Drink size	266	9	12.4
Drink taste	186	8	9.3
Order indication	653	48	51.6
Error indication	558	27	30.7
Telephone number	4374	980	546.8
	<b>31,363</b>	<b>2497</b>	<b>2224.6</b>

system evaluation. For interacting with the system, the simulator employs scenarios that the proposed technique creates from a scenario-templates, as discussed in Section 2, which determine the goals it must try to achieve during the dialogue.

Many dialogue systems employ escalating prompts after an error, i.e., the prompts provide minimal information at first and then add additional prompt phrases if no response is given. The proposed technique allows these prompts. Note, however, that all system's prompts must be answered by the user simulator, which means that it must have available all the necessary data for doing so in the scenarios that the technique creates. Thus, if the system developers decided to create a new prompt type to deal with the error, they should adapt appropriately the simulator, if necessary, so that it can answer the prompt. Moreover, if the data that the simulator would need was not available in the generated scenarios, the developers should modify the scenario-templates to include it. Eventually, this may require the collection of new utterances, for

example, if the new prompt is to get a city name spelled by the user and no utterances of this type are available in the corpus.

## 4. Experiments

The goal of the experiments was to employ the proposed technique to create scenarios and use them to exhaustively evaluate the performance of the Saplen and Viajero systems employing the user simulator discussed in the previous section. As will be discussed in Section 4.2, evaluation results were obtained in terms of word accuracy, concept accuracy, sentence understanding, implicit recovery of speech recognition errors, and task completion. To compute these results we averaged the scores obtained using a 5-fold cross-validation technique. To do so, the sentences in each sentence type shown in Table 1 and Table 2 were divided into five groups, four of them were used for training and the other for evaluation. This process was repeating 5 times, changing the training and evaluation partitions in each iteration.

### 4.1. Utterance corpora and language modelling

The experiments were carried out employing two telephone-based dialogue corpora in Spanish that contain conversations between customers of a fast-food restaurant (Saplen system) and a bus company (Viajero system), which have been used for previous studies, e.g., [27–31]. The corpus for the Saplen system was collected from about 800 dialogues between clients and human telephone operators, and includes product orders, telephone numbers, postcodes, post addresses, queries, confirmations and other types of sentence. The corpus for the Viajero system was collected from around 2200 dialogues between clients and human telephone operators, and includes travel bookings, telephone numbers, queries, confirmations, city name and other types of sentence. Recently, these corpora have been extended with additional utterances taken from the telephone-based dialogues between clients and human operators, as well as from telephone-based dialogues between students of our University and the two systems. All the dialogues were transcribed, labelled and analysed to include tags regarding the speakers (users and telephone operators) and sentence types. The semantic knowledge associated with the sentences was represented in the format of semantic frames, as discussed in Sections 1 and 2. The orthographic transcriptions of the utterances (i.e., the sentences) and their corresponding semantic frames were stored

**Table 2**  
Utterances used for training and evaluating the Viajero system.

Sentence type	Utterances	Time (min)	Size (MB)
Travel booking	2261	308	314.4
Departure times query	632	75	77.9
Arrival times query	533	77	78.9
Travel price query	614	50	53.4
Travel duration query	678	54	58.6
Information needed	440	26	28.8
Error indication	408	20	22.7
Yes/No confirmation	2451	72	85.6
Correction	749	42	46.5
Whish to query about arrival times	435	26	29.4
Whish to query about departure times	509	31	34.1
Whish to query about travel prices	513	27	30.7
Whish to query about travel duration	285	14	16.7
Whish to make travel booking	512	29	32.3
City name	5366	292	319.6
Week day	618	33	37.5
Period of day	324	16	18.3
Telephone number	4017	497	511.6
Hour	1527	77	85.9
Number of seats	590	21	25.9
	<b>23,462</b>	<b>1787</b>	<b>1908.8</b>

in the corpora together with the utterances<sup>3</sup>. Table 1 and Table 2 show details about the corpora for both systems. The duration of the corpus for the Saplen system is around 42 hours of speech (2.2 GB of wav files) whereas that for the Viajero system is around 30 hours (1.9 GB of wav files).

Both dialogue systems employed an HTK-based speech recogniser [32,33], which uses acoustic models previously training with the Spanish Albayzin speech database [34]. In these experiments, the systems employed prompt-dependent language models (PDLMs), concretely, Saplen used 17 PDLMs whereas Viajero employed 16 PDLMs.

To create the PDLMs we compiled specific word bigrams from the orthographic transcriptions associated with the training sentences of each type shown in Table 1 and Table 2, excluding the sentences of type “Error indication”. This is so because a user of the systems can utter an error indication (e.g., “You made a mistake”, “It is incorrect”) at any dialogue state to correct system errors, which makes the systems remove the data obtained in the previous interaction and prompt the user to utter the data again. Hence, the error indications used for training were included in each sentence set employed to compile the word bigrams, which enabled the recognition of error indications at any dialogue state.

#### 4.2. Evaluation measures

The systems were evaluated in terms of word accuracy (WA), concept accuracy (CA), sentence understanding (SU), implicit recovery of speech recognition errors (IR), and task completion (TC). WA evaluates the performance of the speech recogniser and represents the proportion of correctly recognised words, which is computed as follows [35]:

$$WA = (w_t - w_i - w_s - w_d) \times 100 / w_t$$

In this equation,  $w_t$  is the total number of words in the analysed sentences, and  $w_i$ ,  $w_s$  and  $w_d$  are the number of words inserted, substituted and deleted by the speech recogniser, respectively.

CA evaluates the performance of the SLU module and represents the proportion of correctly recognised semantic concepts [36]. A semantic concept is a pair of the form:  $\langle \text{frameSlot} \rangle = \langle \text{value} \rangle$ , which represents a piece of information obtained by the system, for example,  $\langle \text{destinationCity} \rangle = \langle \text{Boston} \rangle$ . This measure is computed similarly as WA but considering semantic concepts instead of words, as observed in the following equation:

$$CA = (C_t - C_i - C_s - C_d) \times 100 / C_t$$

where  $C_t$  is the total number of semantic concepts in the analysed sentences, and  $C_i$ ,  $C_s$  and  $C_d$  are the number of semantic concepts inserted, substituted and deleted by the speech recogniser, respectively.

SU also evaluates the performance of the SLU module and represents the proportion of sentences correctly understood by the system. It is computed as follows:

$$SU = S_u \times 100 / S_t$$

where  $S_u$  is the number of analysed sentences for which the obtained frames are correct and  $S_t$  is the total number of analysed sentences.

Implicit recovery of speech recognition errors (IR) is another evaluation measure for the SLU module [37]. It represents the pro-

portion of incorrectly recognised sentences that are correctly understood by the system and is computed as follows:

$$IR = S_u \times 100 / S_e$$

where  $S_u$  is the number of analysed sentences for which the obtained frames are correct, and  $S_e$  is the number of incorrectly recognised sentences. For example, we say that there is implicit recovery if the user utters “Las Vegas”, the speech recognition result is just “Vegas” and the dialogue system understands that the user has uttered the city name “Las Vegas”.

Task completion (TC) evaluates the performance of the whole system and represents the proportion of successful dialogues, i.e., the percentage of dialogues that end with all the scenario goals being achieved by the user simulator. It is computed as follows:

$$TC = D_c \times 100 / D_t$$

where  $D_c$  is the number of successful dialogues and  $D_t$  is the total number of dialogues. During the user-simulation, cancelled dialogues due to exceeded dialogue length, exceeded number of prompts to get data types, or exceeded number of prompts to confirm data types (see Section 3.1), were not considered successful and thus decreased TC.

#### 4.3. Use of the proposed technique

Employing the proposed technique and our utterance corpora, we designed 4 scenario-templates to create scenarios for evaluating the Saplen system, and 5 scenario-templates to create scenarios for evaluating the Viajero system. These scenario-templates were similar to the ones shown in Fig. 6 and Fig. 7. The effort for creating these scenario-templates was very small as we used our knowledge about typical user goals and system performance for the considered application domains. In these experiments we have dealt with fast-food and bus information applications, and creating the 9 scenario-templates for them just took a few minutes. The really costly process is the creation of the scenarios that will be used for system evaluation, but the proposed technique creates them automatically employing the provided scenario-templates and the utterance corpora. If we were the developers of a system designed for a different application domain, for example, weather forecast information, we may think that a scenario-template should contain a goal representing a user prompting the system for weather information for a city on a certain date. We could assume that the database query the system should perform to provide the requested information would require the city name, state name, and date. Thus, these data should be included in the scenario-template as required goals. Optionally, we could include an out-of-domain goal so that the user simulator provides unexpected input to the system; for example, we could specify a fast-food order when the system prompts for the city name. Hence, creating this scenario-template would take less than one minute.

In the next sections we describe the parameters for dialogue generation cancellation, in-domain and out-of-domain goals, and required goals used in the scenario-templates designed for our experiments.

##### 4.3.1. Parameters for dialogue generation cancellation

We set the parameters of the scenario-templates for dialogue generation cancellation (see Section 2.1.1) as follows. The numMaxDialogueTurns parameter was set to the value of 30 for the Saplen system. The reason is that taking into account the structure of the scenarios used for user simulation as well as the dialogue management strategy of the system, a dialogue without any error correction requires 22 turns in total to make the orders, provide the user data (postcode, telephone number and post address) and answer several system prompts, for example, confirmations for

<sup>3</sup> In our corpora, utterances are stored in .wav files, orthographic transcriptions are stored in .txt files, and semantic frames are stored in .RScore files.

# Section 1: Parameters for dialogue generation cancellation

```
numMaxDialogueTurns = 30
numMaxPromptsToGetDataTypes = 3
numMaxPromptsToConfirmDataTypes = 3
```

# Section 2: In-domain goals

```
utteranceCorpus = saplen goalType = makeFoodPriceQuery utteranceTypes = foodPriceQuery
utteranceCorpus = saplen goalType = makeFoodOrder utteranceTypes = foodOrder
utteranceCorpus = saplen goalType = makeDrinkOrder utteranceTypes = drinkOrder
utteranceCorpus = saplen goalType = provideTelephoneNumber utteranceTypes = telephoneNumber
utteranceCorpus = saplen goalType = providePostalCode utteranceTypes = postalCode
utteranceCorpus = saplen goalType = provideAddress utteranceTypes = address
```

# Section 3: Out-of-domain goals

```
utteranceCorpus = viajero goalType = provideDepartureDay utteranceTypes = departureDay promptType = requestPostalCode probOODResp = 1.0
```

# Section 4: Required goals

```
utteranceCorpus = saplen goalType = provideProductAmount utteranceTypes = amount
utteranceCorpus = saplen goalType = provideIngredients utteranceTypes = ingredients
utteranceCorpus = saplen goalType = provideDrinkSize utteranceTypes = drinkSize
utteranceCorpus = saplen goalType = provideAddressType utteranceTypes = addressType
utteranceCorpus = saplen goalType = provideAddressName utteranceTypes = addressName
utteranceCorpus = saplen goalType = provideBuildingNumber utteranceTypes = buildingNumber
```

Fig. 6. Sample scenario-template to create scenarios for evaluating the Saplen system.

# Section 1: Parameters for dialogue generation cancellation

```
numMaxDialogueTurns = 26
numMaxPromptsToGetDataTypes = 3
numMaxPromptsToConfirmDataTypes = 3
```

# Section 2: In-domain goals

```
utteranceCorpus = viajero goalType = makeSeatBooking utteranceTypes = seatBooking
utteranceCorpus = viajero goalType = provideTelephoneNumber utteranceTypes = telephoneNumber
```

# Section 3: Out-of-domain goals

```
utteranceCorpus = saplen goalType = makeFoodOrder utteranceTypes = foodOrder promptType = requestTelephoneNumber probOODResp = 1.0
utteranceCorpus = saplen goalType = makeDrinkOrder utteranceTypes = drinkOrder promptType = requestDepartureCity probOODResp = 1.0
utteranceCorpus = saplen goalType = providePostalCode utteranceTypes = postalCode promptType = requestDepartureTime probOODResp = 1.0
```

# Section 4: Required goals

```
utteranceCorpus = viajero goalType = provideDepartureCity utteranceTypes = cityName departureCityName
utteranceCorpus = viajero goalType = provideDestinationCity utteranceTypes = cityName destinationCityName
utteranceCorpus = viajero goalType = provideDepartureDay utteranceTypes = weekDay
utteranceCorpus = viajero goalType = provideDepartureTime utteranceTypes = dayTime departureDayTime
utteranceCorpus = viajero goalType = provideNumberOfSeats utteranceTypes = amount
```

Fig. 7. Sample scenario-template to create scenarios for evaluating the Viajero system.

the obtained data and acceptance of price and estimated time for home delivery of products. Considering 30 as an interaction limit means that we permitted  $30 - 22 = 8$  correction turns per dialogue, which we consider an ample margin for the user simulator to correct possible system errors. In other words, the user simulator behaved as a “patient” user, who accepted employing 8 dialogue turns to correct errors before hanging up the telephone.

In the scenario-templates for the Viajero system, the parameter numMaxDialogueTurns was set to the value of 26. The reason is

that taking into account the structure of the scenarios used for the simulation as well as the dialogue management strategy of the system, a dialogue without any error correction requires 18 turns in total considering the travel queries about bus departures, duration and price, travel booking and answering of several system prompts, for example, confirmations for the obtained data. Considering 26 as an interaction limit means that we permitted  $26 - 18 = 8$  correction turns per dialogue. As for Saplen, we consider it an ample margin for the simulator to correct possible system errors. Thus,



**Table 3**  
Scenario-templates to create scenarios in the experiments.

Dialogue system	Scenario-template	Goals			# scenarios
		In-domain	Out-of-domain	Required	
Saplen	STS1	foodOrder drinkOrder telephoneNumber postcode postAddress	-	foodAmount ingredients drinkAmount drinkTaste drinkSize buildingNumber	15,078
	STS2	foodOrder drinkOrder telephoneNumber postcode postAddress	cityName	foodAmount ingredients drinkAmount drinkTaste drinkSize buildingNumber	16,098
	STS3	foodOrder drinkOrder telephoneNumber postcode postAddress	travelDurationQuery	foodAmount ingredients drinkAmount drinkTaste drinkSize buildingNumber	15,078
	STS4	foodOrder drinkOrder telephoneNumber postcode postAddress	dayTime weekDay	foodAmount ingredients drinkAmount drinkTaste drinkSize buildingNumber	15,078
Viajero	STV1	busTravelBooking telephoneNumber	-	departureCity destinationCity weekDay departureTime numberOfSeats	16,098
	STV2	busTravelBooking telephoneNumber	streetName	departureCity destinationCity weekDay departureTime numberOfSeats	16,098
	STV3	busTravelBooking telephoneNumber	foodOrder	departureCity destinationCity weekDay departureTime numberOfSeats	16,098
	STV4	busTravelBooking telephoneNumber	foodOrder drinkOrder	departureCity destinationCity weekDay departureTime numberOfSeats	16,098
	STV5	busTravelBooking telephoneNumber	foodOrder drinkOrder postcode	departureCity destinationCity weekDay departureTime numberOfSeats	16,098

the simulator behaved as the same “patient” user as for the Saplen system.

For both systems, we made experiments setting three different values for the parameters `numMaxPromptsToGetDataTypes` and `numMaxPromptsToConfirmDataTypes`. These values are 3, 4 and 5, which means that the user simulator cancelled the dialogues if it had to give or confirm a data type (e.g., telephone number) more than 3, 4 or 5 times, respectively. These two parameters also allow modelling the “user patience” as they represent how many times the user is willing to provide and confirm a given data before hanging up the telephone (the greater the value, the greater the patience).

#### 4.3.2. In-domain and out-of-domain goals

It can be observed in Fig. 6 that during the creation of scenarios for the Saplen system, the in-domain goals will be taken from the saplen corpus, and that the out-of-domain goals will be extracted from the viajero corpus. Similarly, for the Viajero system (Fig. 7)

the in-domain goals will be taken from the viajero corpus, whereas the out-of-domain goals will be extracted from the saplen corpus.

Table 3 sets out the goals included in the scenario-templates as well as the number of different scenarios that will be created using the templates. It also shows the required goals, which will be discussed in Section 4.3.3. Note that the scenario-templates STS1 and STV1 contain in-domain goals only. This is so because with these templates we want to evaluate the dialogue systems in the best case, i.e., considering utterances taken from the correct corpus only. The other scenario-templates contain both in-domain and out-of-domain goals because with them we want to evaluate the system’s ability in dealing with the ASR, SLU and DM errors caused by the processing of utterances taken from the incorrect corpus.

The last column of the table shows the total number of scenarios that will be created employing the three values for the `numMaxPromptsToGetDataTypes` and `numMaxPromptsToConfirmDataTypes` parameters (i.e., 3, 4 and 5). For example, for the scenario-template STS1 we will create  $5026 \times 3 = 15,078$  scenarios for each value of these parameters, thus  $5026 \times 3 = 15,078$  scenarios.

#### 4.3.3. Required goals

The scenario-templates STS1 – STS4 of Table 3 are concerned with product orders of a fast-food restaurant and will be used to create scenarios for evaluating the Saplen system. These templates contain all the goals that are necessary for fully registering food and drink orders as well as for registering user data for home delivery of the ordered products (i.e., telephone number, postcode and post address). Since we include a food order in the in-domain section of the scenario-templates, we need to include goals for the food amount and the ingredients in the section for required goals. The reason is that there are utterances in the saplen corpus in which the speakers did not mention both data (e.g., “ham sandwich”, “please I would like one sandwich”) and these data are mandatory for the system to fully register food orders.

Similarly happens with the inclusion of the drink order in the section for in-domain goals. We need to include goals for the drink amount, drink size and drink taste in the section for required goals, since there are utterances in the corpus in which the speakers did not mention all of these data (e.g., “large Fanta”, “one milkshake”, “one Fanta”) and these data are mandatory for the system to fully register drink orders.

Due to the same reason, we need to include a goal for the building number in the section for required goals, since there are utterances of post addresses in the corpus in which the speakers did not mention the building number (e.g., “Elm street, second floor, letter C”) and this data is mandatory for the system to fully register the user's post address.

The scenario-templates STV1 – STV5 of Table 3 are concerned with the bus travel bookings that we will use to create scenarios for evaluating the Viajero system. These scenario-templates contain all the goals that are necessary for fully registering bus travel bookings, i.e., travel booking and the user's telephone number. Since we include a travel booking in the in-domain section of the scenario-templates, we need to include goals for the departure and destination cities, week day, departure time, and number of seats in the section for the required goals. The reason is that there are bookings in the viajero corpus that are incomplete because the speakers did not mention all of these data, which are necessary for the system to fully register a booking. For example, in the sentence “Please I want to go to Madrid leaving from Granada this Friday” the number of seats is missing.

#### 4.3.4. Creation of scenarios

Employing the proposed technique and the scenario-templates discussed in the previous section, we created the number of scenarios indicated in the last column of Table 3. We employed all the possible semantic frames in the corpora used for evaluation to consider an ample range of goals. The simulator's selection of utterances to answer systems' prompts was carried out considering that all the utterances of a given type should be used the same number of times. Thus, if during the interaction with a dialogue system the simulator had to answer a system prompt with an affirmative confirmation (e.g., “yes”), it searched for the least used utterance associated with an affirmative confirmation.

Fig. 8 shows a sample scenario for evaluating the Saplen system, in which the scenario goals are marked in bold to ease reading. It can be observed that the in-domain goals are concerned with asking for the price of a ham sandwich, ordering a sandwich and an orange Fanta, and finally providing a telephone number, a postcode and a post address. The scenario contains an out-of-domain goal, which will make the user simulator to provide with probability 1.0 the week day “Thursday” the first time the system prompts for the postcode. To do this, the simulator will take from the viajero corpus a sentence recorded by a speaker who uttered that week day.

Using the Saplen system, the fully making of a drink order requires that the user specifies the amount, size (either ‘normal’ or ‘large’) and taste (e.g., ‘lemon’, ‘orange’). Note, however, that the size data is not in the scenario's semantic frame used for making the order. This is so because the fast-food restaurant's client who was recorded to collect the saplen corpus just uttered “one orange Fanta please”. Hence, to enable the user simulator in fully making the order, the scenario includes the drink size data in the section for required goals.

Fig. 9 shows a sample scenario for evaluating the Viajero system. It can be observed that the in-domain goals are concerned with booking a bus travel to go from Madrid to Barcelona on a direct bus, getting information on travel duration and price, and providing a telephone number. The scenario contains an out-of-domain goal, which will make the user simulator to provide with probability 1.0 the street name “Velazquez” the first time the system prompts for the telephone number. To do this, the simulator will take from the saplen corpus a sentence that was recorded by a fast-food restaurant's client uttering that street name.

Using the Viajero system, the fully making of a travel booking requires that the user specifies, in addition to their telephone number, the departure and destination cities, the week day and time of departure, and the number of seats to be reserved. Note, however, that the data regarding departure day, departure time and number of seats to be reserved are not in the scenario's semantic frame used for making the booking. This is so because the bus company's customer who was recorded to collect the viajero corpus just uttered “Please I would like to make a booking from Madrid to Barcelona on a direct bus”. Hence, to enable the user simulator in fully making the booking, the scenario includes these data in the section for required goals.

#### 4.4. User-simulation experiments with the dialogue systems

As discussed in Section 3.1, making experiments with real users interacting with spoken dialogue systems takes a lot of time and effort. Hence, instead of employing real users, we carried out experiments employing a user-simulation technique. The goal of these experiments was not only to check the performance of the systems' ASR and SLU modules, but also that of the DM module. The behaviour of the DM module is critical for a satisfactory user interaction, since among other tasks, the module must employ appropriate dialogue management strategies so that the user can obtain the requested service or information as efficiently as possible. To do so, the DM is responsible for initiating error handling sub dialogues, which mostly try to recover the system from speech recognition and/or spoken language understanding errors. Fig. 10 shows a sample dialogue<sup>4</sup> translated from Spanish into English showing the Saplen system and our user simulator described in Section 3.1.

This dialogue was generated from the sample scenario shown in Fig. 8. It can be observed that in the dialogue turn (6) the user simulator ordered a drink but did not specify the drink size, which is necessary for fully registering the order. This happened because the goal in the scenario (see Fig. 8) did not include this data. This made the dialogue system to prompt for this data in the dialogue turn (7). In the turn (8) the simulator provided the size “large”, in accordance with the goal in the required goals section of the scenario. In the dialogue turn (16) the simulator provided a telephone number, and in the turn (17) the system generated a prompt to confirm it. The simulator provided a negative confirmation in the

<sup>4</sup> In this dialogue the speech recognition hypotheses include within brackets confidence scores in the range 0 – 1.0 attached to the recognised words. According to our experimental setting, the dialogue system confirms a recognised word if its confidence score is lower than 0.5.

```

# Section 1: Parameters for dialogue generation cancellation

numMaxDialogueTurns = 30
numMaxPromptsToGetDataTypes = 3
numMaxPromptsToConfirmDataTypes = 3

# Section 2: In-domain goals

# ../utteranceCorpora/saplen/foodPriceQuery_20121214_13-24-43.RScor useCount = 10 utteranceCorpus =
saplen goalType = makeFoodPriceQuery
<action> = "foodPriceQuery"
<amount> = "1"
<food> = "sandwich"
<ingredients> = "ham"

# ../utteranceCorpora/saplen/foodOrder_20070509_12-28-35.RScor useCount = 25 utteranceCorpus = saplen
goalType = makeFoodOrder
<action> = "foodOrder"
<amount> = "1"
<food> = "sandwich"
<ingredients> = "ham"

# ../utteranceCorpora/saplen/drinkOrder_20070509_12-28-35.RScor useCount = 15 utteranceCorpus = saplen
goalType = makeDrinkOrder
<action> = "drinkOrder"
<amount> = "1"
<drink> = "fanta"
<taste> = "orange"

# ../utteranceCorpora/saplen/telephoneNumber_20040312_19-52-39.RScor useCount = 1 utteranceCorpus =
saplen goalType = provideTelephoneNumber
<telephoneNumber> = "958275360"

# ../utteranceCorpora/saplen/postcode_20080501_12-25-19.RScor useCount = 12 utteranceCorpus = saplen
goalType = providePostcode
<postcode> = "18014"

# ../utteranceCorpora/saplen/postAddress_20090607_09-34-54.RScor useCount = 1 utteranceCorpus = saplen
goalType = providePostAddress
<postAddressType> = "street"
<streetName> = "Andalucia"
<buildingNumber> = "58"
<buildingFloor> = "1"
<apartmentLetter> = "b"

# Section 3: Out-of-domain goals

# ../utteranceCorpora/viajero/weekDay_20081112_20-39-55_NOR.RScor UseCount = 1 utteranceCorpus =
viajero goalType = provideDepartureDay systemPrompt = requestPostcode probOODResp = 1.0
<departureDay> = "Thursday"

# Section 4: Required goals

# ../utteranceCorpora/saplen/drinkSize_20070408_12-28-35.RScor useCount = 13 utteranceCorpus = saplen
goalType = provideDrinkSize
<size> = "large"

```

Fig. 8. Sample scenario for evaluating the Saplen system.

turn (18) since the number was incorrectly understood, because the word “fifty\_three” was recognised as “sixty\_three”. The system generated a second prompt to confirm the number because the confidence score of the word “fifty\_three” (0.4982) was lower than the confidence threshold, and in the turn (22) the simulator provided an affirmative confirmation. It should be noted that getting the telephone in this dialogue required a total of eight turns,

from the turn (15) to the turn (22). In the dialogue turn (23) the system prompted for a postcode, but in the turn (24) the simulator used the out-of-domain goal of the scenario and provided a week day instead of a postcode. The simulator’s response was recognised as “go back”, which made the dialogue system to confirm the last obtained data (telephone number) and prompt again for the postcode. It should be noted that getting the postcode required a total

```

# Section 1: Parameters for dialogue generation cancellation

numMaxDialogueTurns = 26
numMaxPromptsToGetDataTypes = 3
numMaxPromptsToConfirmDataTypes = 3

# Section 2: In-domain goals

# ../utteranceCorpora/viajero/TravelBooking_20100327_15-01-49.RScor useCount = 11 utteranceCorpus =
viajero goalType = makeTravelBooking
<action> = "TravelBooking"
<departureCity> = "Madrid"
<destinationCity> = "Barcelona"
<busType> = "direct"

# ../utteranceCorpora/viajero/travelDurationQuery_20071229_16-26-48.RScor useCount = 17 utteranceCorpus =
viajero goalType = makeTravelDurationQuery
<action> = "travelDurationQuery"

# ../utteranceCorpora/viajero/travelPriceQuery_20030916_20-16-32.RScor useCount = 9 utteranceCorpus =
viajero goalType = makeTravelPriceQuery
<action> = "travelPriceQuery"

# ../utteranceCorpora/viajero/telephoneNumber_20020126_18-17-20.RScor useCount = 1 utteranceCorpus =
viajero goalType = provideTelephoneNumber
<telephoneNumber> = "696812896"

# Section 3: Out-of-domain goals

# ../utteranceCorpora/saplen/streetName20091229_12-19-24.RScor useCount = 4 utteranceCorpus = saplen
goalType = provideStreetName promptType = requestTelephoneNumber probOODResp = 1.0
<streetName> = "velazquez"

# Section 4: Required goals

# ../utteranceCorpora/viajero/weekDay_20120421_12-36-56.RScor useCount = 26 utteranceCorpus = viajero
goalType = provideDepartureDay
<weekDay> = "Friday"

# ../utteranceCorpora/viajero/dayTime_20091211_22-04-36.RScor useCount = 13 utteranceCorpus = viajero
goalType = provideDepartureTime
<time> = "10:00"

# ../utteranceCorpora/viajero/amount_20010517_12-09-36.RScor useCount = 21 utteranceCorpus = viajero
goalType = provideNumberOfSeats
<amount> = "2"

```

Fig. 9. Sample scenario for evaluating the Viajero system.

of six dialogue turns, from the turn (23) to the turn (28). The dialogue generation was cancelled by the simulator because the `numMaxDialogueTurns` parameter (set to the value 30) was exceeded due to the additional turns employed to get the telephone number and the postcode.

It should also be noted that our simulator can employ utterances from different speakers, which is not realistic. For example, one dialogue may contain utterances of users who might be either calmed down or frustrated. However, doing so allows a large flexibility in the generation of simulated dialogues. For example, the proposed technique can create a scenario containing the goals `<telephoneNumber> = "958123456"` and `<postCode> = "18014"`, even though both goals are not uttered by the same speaker. Moreover, it should be mentioned that the utterances in our corpora are tagged reflecting the emotional state of the speakers (either "Normal", "Angry" or "Tired"). In this way,

our user simulator always selects the utterances considering the appropriate emotional state so that the simulated dialogues look realistic. Concretely, the simulator starts the dialogues employing utterances of emotional state "Normal", and changes to utterances of emotional state "Angry" or "Tired" when it has to repeat either data or confirmations due to problems in the dialogue system's performance.

Our simulator excludes over-generation of dialogues since the proposed technique creates the number of scenarios necessary to use, just once, each utterance from the largest set of utterance types representing goals in the scenario-template. For example, let us consider that the scenario-template specifies to include in the scenarios goals representing telephone numbers and city names only. Also, assume that in the utterance corpus there are 4000 utterances of telephone numbers and 5000 utterances of city names. Thus, our technique would create 5000 scenarios to use just once



- (1) **Saplen:** Good evening, welcome to our restaurant. This is our automatic telephone service system. The minimum price for the home delivery service is five euros. Please speak after the tone. What would you like to order?
  - (2) **User simulator:** What is the price of a ham sandwich [Recognition hypothesis: what (0.7192) is (0.9726) the (0.7351) price (0.8261) of (0.5781) a (0.5926) ham (0.7928) sandwich (0.9261)]
  - (3) **Saplen:** A ham sandwich costs 3 euros. What would you like to order?
  - (4) **User simulator:** One ham sandwich [Recognition hypothesis: one (0.6285) ham (0.5956) sandwich (0.8510)]
  - (5) **Saplen:** OK. One ham sandwich. Would you like to order a drink?
  - (6) **User simulator:** One orange Fanta [Recognition hypothesis: one (0.5698) orange (0.8361) fanta (0.9764)]
  - (7) **Saplen:** OK. One orange fanta. What size would you like?
  - (8) **User simulator:** Large [Recognition hypothesis: large (0.5969)]
  - (9) **Saplen:** OK. So that's one large orange Fanta. Would you like anything else?
  - (10) **User simulator:** No [Recognition hypothesis: no (0.5972)]
  - (11) **Saplen:** I need to confirm your order. I think you ordered one ham sandwich and one large orange Fanta. Is this correct? Please answer yes or no.
  - (12) **User simulator:** Yes [Recognition hypothesis: yes (0.6384)]
  - (13) **Saplen:** OK. The price is 5 euros. Is this OK? Please answer yes or no.
  - (14) **User simulator:** Yes [Recognition hypothesis: yes (0.5821)]
  - (15) **Saplen:** OK. Please say your telephone number.
  - (16) **User simulator:** 9 5 8 27 53 60 [Recognition hypothesis: nine (0.6851) five (0.6985) eight (0.7659) twenty\_seven (0.7565) sixty\_three (0.2642) sixty (0.7521)]
  - (17) **Saplen:** I need to confirm your telephone number. Did you say 9, 5, 8, 27, 63, 60? Please answer yes or no.
  - (18) **User simulator:** No [Recognition hypothesis: no (0.6834)]
  - (19) **Saplen:** OK. Please say your telephone number again.
  - (20) **User simulator:** 9 5 8 27 53 60 [Recognition hypothesis: nine (0.6091) five (0.6952) eight (0.7928) twenty\_seven (0.8082) fifty\_three (0.4982) sixty (0.6927)]
  - (21) **Saplen:** I need to confirm your telephone number. Did you say 9, 5, 8, 27, 53, 60? Please answer yes or no.
  - (22) **User simulator:** Yes [Recognition hypothesis: yes (0.7341)]
  - (23) **Saplen:** OK. Telephone number 9, 5, 8, 27, 53, 60. Please say your postcode.
  - (24) **User simulator:** Thursday [Recognition hypothesis: go (0.8561) back (0.7637)]
  - (25) **Saplen:** I got your telephone number is 9, 5, 8, 27, 53, 60. Is this correct? Please answer yes or no.
  - (26) **User simulator:** Yes [Recognition hypothesis: yes (0.9671)]
  - (27) **Saplen:** OK. Please say your postcode again.
  - (28) **User simulator:** 18, 0, 14 [Recognition hypothesis: eighteen (0.6781) zero (0.8591) fourteen (0.6106)]
  - (29) **Saplen:** OK. Postcode 18, 0, 14. Please state your post address saying a place name for the letter of the apartment, for example Elm street, number 13, second floor, letter b for Barcelona.
  - (30) **User simulator:** Andalucia street number 58 first floor letter b for Barcelona [Recognition hypothesis: andalucia (0.8659) street (0.76395) number (0.5963) fifty-eight (0.9683) first (0.9465) floor (0.8029) letter (0.7869) b\_for\_Barcelona (0.5361)]
- \*\* DIALOGUE GENERATION CANCELLED BY USER SIMULATOR DUE TO EXCESSIVE LENGTH \*\***

**Fig. 10.** Sample dialogue between the user simulator and the Saplen system.

each city name. This means that 4000 telephone numbers would be used just once and 1000 numbers would be used twice.

It is also worth mentioning that all the simulated dialogues are valid in the context of the considered application domains. As discussed in Section 4.2, a dialogue ends successfully only if the user simulator has achieved all the goals in the scenario, which means task completion (TC). If the dialogue takes too many turns or the user simulator has to provide (or confirm) data too many times, the simulator cancels the dialogue to replicate that a real user in this case would hang up the telephone.

#### 4.4.1. Results obtained with Saplen

Fig. 11 shows the average scores obtained from the analysis of the dialogues generated from the interaction between the user simulator and the Saplen system. To generate the dialogues we employed the scenarios that we created employing the scenario-templates STS1 - STS4 shown in Table 3.

In this figure, “UP” denotes “User patience”, which refers to the parameters numMaxPromptsToGetDataTypes and numMaxPromptsToGetDataConfirmed that were discussed in Section 4.3.1 (note that for simplicity, in these experiments we have given the same value to both parameters: either 3, 4 or 5). It can be observed in the figure that the best results were obtained for the scenario-template STS1 and UP = 5, for which we attained WA = 88%, CA = 84%, SU = 82%, IR = 2% and TC = 64%. These scores show that better results are obtained if we employ the scenarios created using the scenario-template that contains in-domain goals only (see Table 3) because in this case the dialogue system did not need to deal with utterances taken from the viajero corpus. The value of UP did not have an impact on the WA, CA, SU and IR results. However, it had an impact on TC regardless of the scenario-template. This result shows that the greater the value of the User Patience parameters, the greater the number of attempts for the dialogue

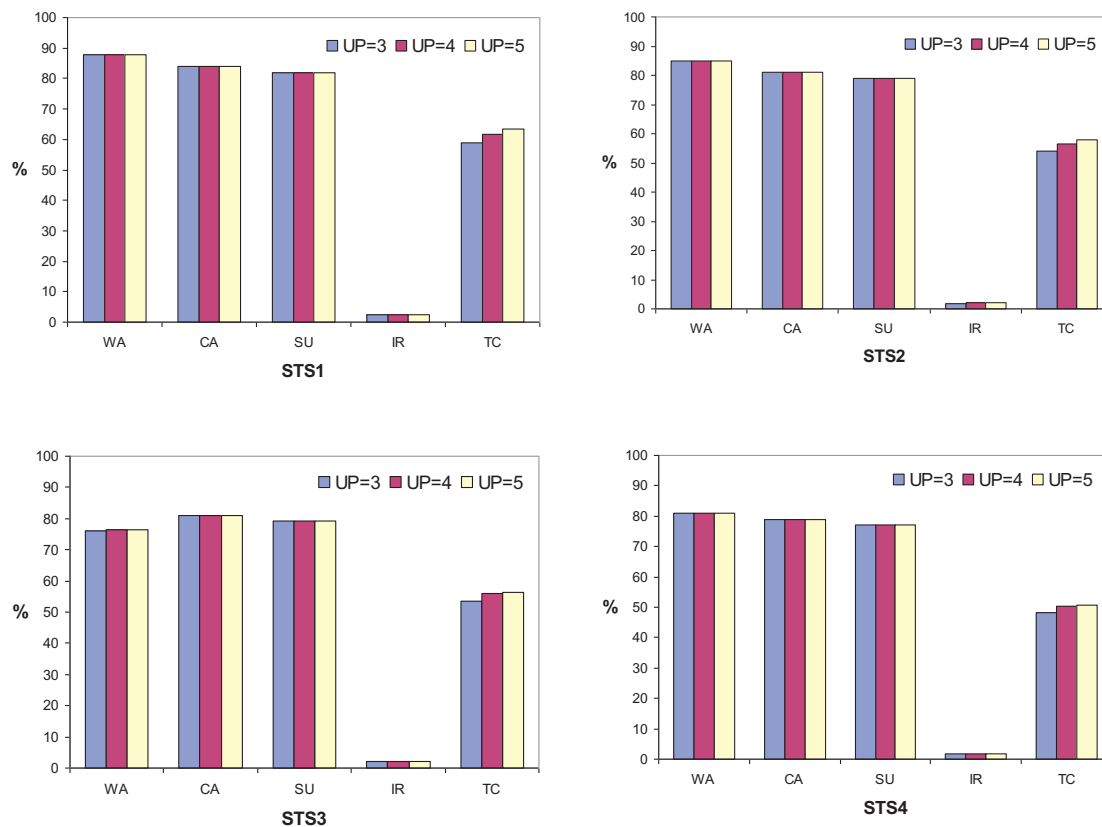


Fig. 11. User-simulation average results obtained with Saplen for the scenario-templates STS1 - STS4.

Table 4

Dialogue turns and words uttered by the user simulator interacting with Saplen.

Scenario-template	UP	# Simulator's turns	# System's turns	# Simulator's words
STS1	3	103,110	108,136	289,000
	4	110,714	115,740	305,928
	5	114,556	119,582	314,225
STS2	3	118,534	123,900	318,091
	4	126,154	131,520	335,082
	5	129,210	134,576	341,250
STS3	3	111,396	116,422	335,860
	4	118,390	123,416	350,420
	5	121,546	126,572	359,011
STS4	3	119,061	124,087	322,346
	4	126,000	131,026	337,721
	5	129,008	134,034	343,034

system to get the data from the user simulator and confirm the data, which obviously helps to achieve all scenario goals.

The worst results were obtained for the scenario-template STS4 and  $UP = 3$ , for which we attained  $WA = 81\%$ ,  $CA = 79\%$ ,  $SU = 77\%$ ,  $IR = 2\%$  and  $TC = 48\%$ . This happens because the scenarios created using this scenario-template contain two out-of-domain goals, whereas the scenarios created with the templates STS2 and STS3 just contain one (see Table 3). Hence, it was more difficult for the dialogue system to achieve all the goals in the scenarios created using STS4, which lead to the smallest TC score.

Table 4 shows the scores concerned with the total number of dialogue turns by the user simulator and the Saplen system, as well as the total number of words uttered by the simulator. As can be observed, the greater the User Patience (UP) for each scenario-template, the greater the number of dialogue turns and uttered words. The lowest scores were obtained for the scenario-template STS1 since in this case the user simulator did not need to utter words regarding out-of-domain goals, and consequently the dia-

logue system did not need to employ additional dialogue turns for recovering from errors caused by the misrecognition of these words.

Table 5 sets out the results of an analysis of the generated dialogues in which we looked for the main reasons for no task completion. The third column shows the number of dialogues that did not end successfully because the value of the parameter `numMaxDialogueTurns` was exceeded. The other columns show the number of dialogues that did not end successfully because the value of the User Patience (UP) was exceeded when the system tried to get and confirm the telephone number, street name, post-code, building number and apartment letter, respectively. As can be observed, the main reason for no task completion regardless of the scenario-template and the value of UP is that the maximum number of dialogue turns was exceeded. This mostly happened because of the dialogue strategy that the system followed to confirm the user's post address. According to this strategy, when the system prompted for the address the user typically uttered all the

**Table 5**

Main reasons for no task completion using Saplen.

Scenario-template	UP	Number max. dialogue turns	Telephone number	Street name	Postcode	Building number	Apartment letter
STS1	3	1489	333	297	237	102	74
	4	1394	198	197	83	48	21
	5	2197	97	107	37	20	4
STS2	3	1152	319	307	213	126	117
	4	1583	195	216	91	49	51
	5	1855	121	132	38	24	12
STS3	3	1215	353	331	238	132	112
	4	1680	213	227	93	60	49
	5	1892	115	156	39	23	14
STS4	3	873	307	301	207	123	160
	4	1319	192	206	81	48	61
	5	1495	99	158	32	22	19

data at a time, and in the next dialogue turn the system prompted the user to confirm all the data at a time. If the user uttered a negative confirmation, the system started an error recovery sub dialogue, prompting the user to confirm each individual data in the address (e.g., the street name). Moreover, the system prompted whether optional data which were not obtained from the user should actually be in the address (e.g., the building floor). If the user answered yes to this question, the system then prompted for these data and asked the user to confirm them. In the end of this recovery sub dialogue, it was very likely that the system could count on all the data in the user address properly confirmed. The problem with this strategy is that it needed many additional dialogue turns, which caused the dialogues to be much longer.

The second mostly observed reason for no task completion is the system's difficulty in getting the user's telephone number. The reason is that getting the number correctly understood requires recognising correctly the whole sequence of digits and/or pairs of digits that make up the number. Hence, if at least one digit or pair of digits is incorrectly recognised, the telephone number is incorrectly understood by the system. We observe that this problem occurs often in our experiments due to the acoustic similarity between some Spanish words, e.g., "sesenta" (sixty) and "setenta" (seventy). Also, there were ASR problems caused by the strong southern Spanish accents of most speakers in the saplen corpus, which changed the pronunciation of some numbers. For example, we observed that "dos" (two) was often recognised as "ocho" (eight) since most speakers did not pronounce the final 's' of "dos".

We observed the same problems in getting the user's postcode, since to correctly obtain a postcode the system must obtain correctly the sequence of digits and/or pairs of digits that make up the code. Hence, if there is at least one error, the postcode is incorrectly recognised.

Regarding the other cases for no task completion, we observed again ASR errors due to the acoustic similarity between some words and the strong Southern Spanish accents of most speakers in the utterance corpora. For example, we found that the apartment letter "b de Belgrado" (b for Belgrade) was sometimes recognised as "d de dado" (d for dice) due to the acoustic similarity, and that the city name "muriendas" was often recognised as "ubierna" since it was often pronounced without the final 's'.

#### 4.4.2. Results obtained with Viajero

Fig. 12 shows the average scores obtained from the analysis of the dialogues generated from the interaction between the user simulator and the Viajero system. To generate the dialogues we employed the scenarios that we created employing the scenario-templates STV1 - STV5 shown in Table 3. In this figure "UP" denotes "User Patience" as discussed for the Saplen system. It can

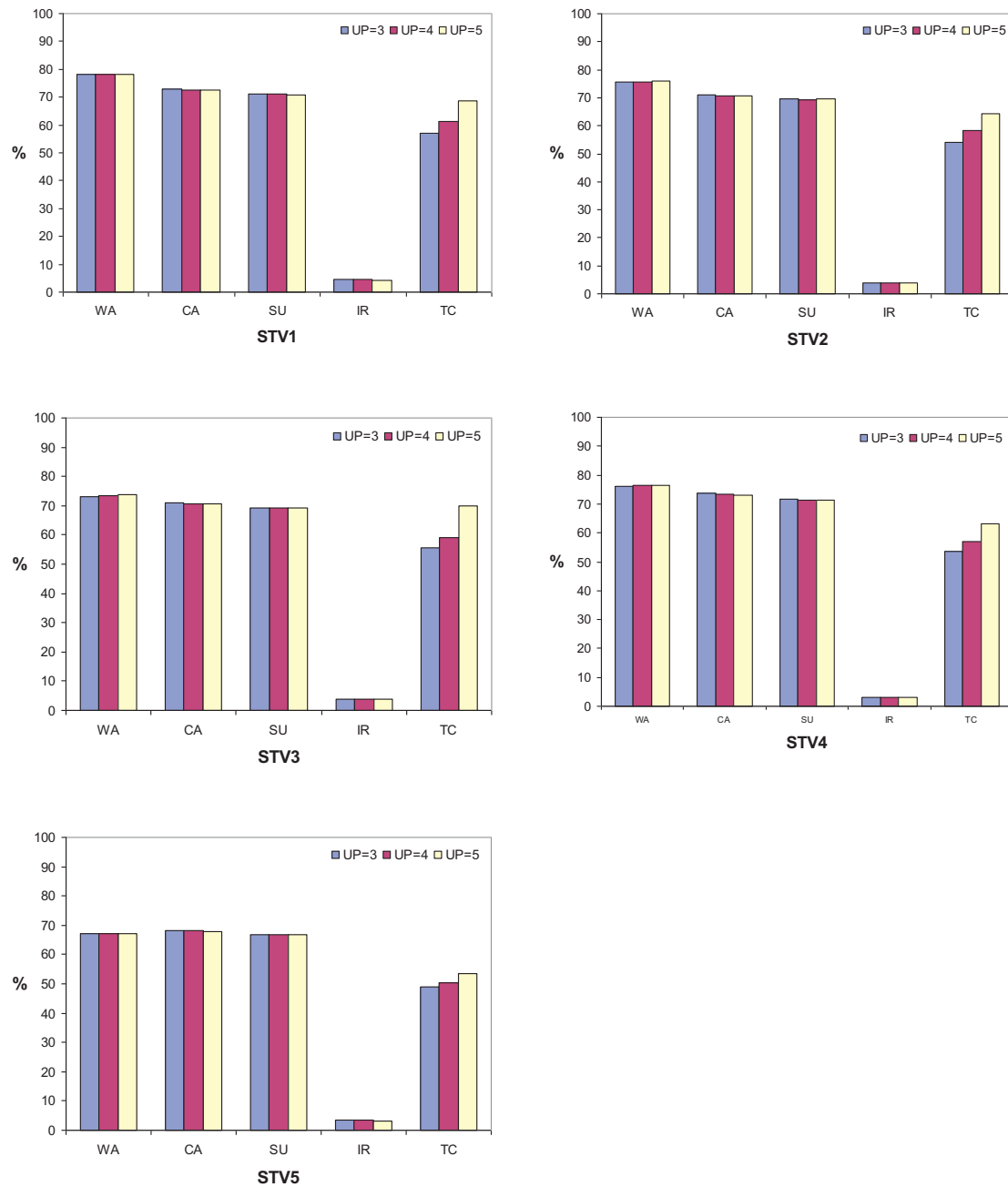
be observed in the figure that the best results were obtained for the scenario-template STV1 and UP=5, for which we attained WA = 78%, CA = 72%, SU = 71%, IR = 4% and TC = 69%. These scores show that better results are obtained if we employ the scenario-template that contains in-domain goals only (see Table 3) because in this case the dialogue system did not need to deal with utterances taken from the saplen corpus. As for Saplen, the value of UP did not have an impact on the WA, CA, SU and IR results. However, it had an impact on TC regardless of the scenario-template. This result shows that the greater the value of the User Patience parameters, the greater the number of attempts for the dialogue system to get the data from the user simulator and confirm the data, which obviously helps to achieve all scenario goals.

The worst results were obtained for the scenario-template STV5 and UP = 3, for which we attained WA = 67%, CA = 68%, SU = 67%, IR = 3% and TC = 49%. This happens because the scenarios created using this scenario-template contain three out-of-domain goals, whereas the scenarios created with the templates STV2 and STV3 just contain one, and those created with the template STV4 contain two (see Table 3). Hence, it was more difficult for the dialogue system to achieve all the goals in the scenarios created using STV5, which lead to the smallest TC score.

Table 6 shows the scores concerned with the total number of dialogue turns by the user simulator and the Viajero system, as well as the total number of words uttered by the simulator. As can be observed, the greater the User Patience (UP) for each scenario-template, the greater the number of dialogue turns and uttered words. The lowest scores were obtained for the scenario-template STV1 since in this case the user simulator did not need to utter words regarding out-of-domain goals, and consequently the dialogue system did not need to employ additional dialogue turns for recovering from errors caused by the misrecognition of these words.

Table 7 sets out the results of an analysis of the generated dialogues with the Viajero system, in which we looked for the main reasons for no task completion. As for Saplen, the third column shows the number of dialogues that did not end successfully because the value of the parameter numMaxDialogueTurns was exceeded. The other columns show the number of dialogues that did not end successfully because the value of the User Patience (UP) was exceeded when the system tried to get and confirm the telephone number, city name, and day time.

The table shows that the main reason for no task completion regardless of the type of scenario-template is the system's problems in fully getting the city name, either departure or destination. The reason is similar as that discussed for the Saplen system regarding the street names, i.e., acoustic similarity between some uttered city names. For example, we observed that the city name "luque" was recognised as "rute". Also, there were ASR problems caused by the strong southern Spanish accents of most speakers



**Fig. 12.** User-simulation average results obtained with Viajero for the scenario-templates STV1 – STV5.

**Table 6**

Dialogue turns and words uttered by the user simulator interacting with Viajero.

Scenario-Template	UP	# Simulator's turns	# System's turns	# Simulator's words
STV1	3	87,738	93,022	318,802
	4	94,530	99,781	337,695
	5	101,967	101,237	353,518
STV2	3	100,656	105,947	341,749
	4	106,746	112,028	357,178
	5	113,549	118,821	372,757
STV3	3	99,049	104,326	355,425
	4	106,302	111,586	373,620
	5	113,107	118,378	389,547
STV4	3	94,058	99,339	354,518
	4	101,164	106,445	374,405
	5	108,404	113,679	391,915
STV5	3	115,047	120,333	417,006
	4	121,864	127,145	433,274
	5	128,178	133,458	450,503



**Table 7**

Main reasons for no task completion using Viajero.

Scenario-template	UP	Number max. dialogue turns	Telephone number	City name	Day time
STV1	3	152	479	967	194
	4	227	381	913	132
	5	384	245	593	62
STV2	3	326	414	1018	196
	4	436	290	992	117
	5	707	171	593	52
STV3	3	248	442	1001	183
	4	398	326	952	122
	5	655	219	595	53
STV4	3	278	527	952	214
	4	460	391	899	133
	5	680	265	584	59
STV5	3	539	461	1023	166
	4	860	308	951	89
	5	1278	232	567	25

in the viajero corpus, which changed the pronunciation of many words. For example, we observed that the city name “sabinillas” was recognised as “pardilla” since the speakers did not pronounce the final ‘s’ of “sabinillas”.

The second main reason for no task completion is the system’s problem in fully getting the user’s telephone number, which happened due to the same reasons discussed for the Saplen system, i.e., difficulty in correctly getting all the digits and/or pairs of digits that make up a telephone number in the format typically used in Spain, due to the acoustic similarity between some uttered words, and the strong southern Spanish accents of most speakers in our utterance corpora.

The third main reason for no task completion is that the maximum number of dialogue turns was exceeded. This happened especially because of problems in fully getting all the data concerned with bus travel bookings (i.e., departure city, destination city, week day, number of seats, and departure time). According to the system’s dialogue strategy for dealing with the bookings, when the user enters a booking the system informs of the obtained data and prompts for a confirmation of all the data at a time. If the user enters a negative confirmation, the system prompts for one wrong data (e.g., destination city). Then, the system prompts for the data, asks for a confirmation, and again prompts for a confirmation of all the data at a time. This process is repeated until the user enters an affirmative confirmation. Hence, in the end of this procedure it is very likely that the system can count on all the correct data to send the booking data to the database. The problem with this strategy is that it requires additional dialogue turns, which makes the dialogues longer, causing some of these to exceed the maximum number of dialogue turns. However, this strategy requires less dialogue turns than the one employed by Saplen in fully getting the data concerned with the user’s post address.

The fourth reason for no task completion is concerned with the system difficulty in getting the day time, due again to the acoustic similarity between some uttered words, especially when uttered by strongly accented speakers. For example, we observed that the time expression “en torno a las seis” (at around six) was sometimes recognised as “en torno a las diez” (at around ten), and that the expression “a las once y media” (at half past eleven) was sometimes recognised as “a las doce y media” (at half past twelve).

#### 4.4.3. Summary of results

Table 8 shows the absolute differences for the evaluation measures considering the best and worst cases for both systems. As can be observed, the differences are very noticeable for WA, CA and SU. For example, WA is 7% higher (88% vs. 81%) for Saplen and 11% higher (78% vs. 67%) for Viajero. Most importantly, the difference is very remarkable for TC, which is 16% higher (64% vs. 48%) for

**Table 8**

Absolute differences between the best and worst cases for both systems.

System	WA	CA	SU	IR	TC
Saplen	7%	5%	5%	0%	16%
Viajero	11%	4%	4%	1%	20%

Saplen and 20% higher (69% vs. 49%) for Viajero. These differences show that considering (or not) out-of-domain goals has a significant impact on the systems’ ability in recognising and understanding the sentences uttered by the users. Moreover, the simulation of users who are more (or less) patient in dealing with system problems has a significant impact on the systems’ ability to provide the service requested by the users. The combination of these two factors explains the very remarkable difference in terms of TC. However, the difference is very small for IR, which shows that these changes in the configuration used for dialogue generation do not have a significant impact of the systems’ ability in recovering from ASR errors.

It should be noted that the differences reported in the table are obtained from the analysis of a large number of dialogues that we generated employing the scenarios created using the proposed technique. Concretely, we employed 10,052 dialogues generated with Saplen and the 10,732 dialogues generated with Viajero. The generation of these dialogues required several full-days of intense work of four i7 Intel 3.40 GHz 16GB PCs running simultaneously and employing the 54,825 utterances in our corpora.

## 5. Conclusions and future work

In this paper we have presented a novel technique to create scenarios that can be used by a user simulator for exhaustively evaluating spoken dialogue systems. To the best of our knowledge, this is the first paper in proposing a technique to create these scenarios. According to our technique, the scenarios are automatically created from simple scenario-templates that the systems’ developers create manually employing their knowledge about typical goals of the system’s users. The scenarios contain goals, which the simulator will try to achieve through the interaction with the systems. The goals are represented in the form of semantic frames, which are associated with user utterances of sentences and are taken from utterance corpora. In this way, the scenarios enable speech-based interaction between the simulator and the spoken dialogue systems to be evaluated. The technique considers three sets to model the knowledge of the systems’ developers about the activity of the users (set of goal types  $G_T$ , set of goals  $G$ , and set of

utterance types  $U_T$ ), and a set to model the knowledge of the systems' developers about the activity of the dialogue system to be evaluated (set of prompt types  $P_T$ ).

We have considered two types of goal that we call *in-domain* and *out-of-domain* goals. The former are taken from an utterance corpus representative of the dialogue system's application domain, whereas the latter are taken from utterance corpora not concerned with the system's application domain. Using the latter, it is possible to evaluate the system's robustness in dealing with unexpected sentences, and thus its ability in recovering from the errors that these goals will cause. Unlike other evaluation techniques based on user simulation, the proposed technique allows evaluating system performance employing real speech. Hence, to some extent, it makes the simulation more similar to that that could be carried out with real users, since the dialogue systems must face real, not simulated ASR problems.

Experiments were carried out employing two spoken dialogue system previously developed in our lab, one designed for the fast-food ordering domain (Saplen system) and the other for the bus travel information domain (Viajero system). Both systems employed an HTK-based speech recogniser that used prompt-dependent language models (word bigrams) compiled from utterance corpora corresponding to the two domains. For each system, we employed the proposed technique and created scenarios considering several scenario-templates, one of them containing in-domain goals only, and the others containing both in-domain and out-of-domain goals. The scenarios were employed by a user-simulator to interact with the dialogue systems considering three different values (3, 4 and 5) for the two parameters that modelled the "User Patience" (UP). As a result of the user simulation, we obtained a total of 141,822 dialogues that were analysed in terms of word accuracy, concept accuracy, sentence understanding, implicit recovery of speech recognition errors, and task completion.

The results of the analysis set out in Fig. 11 and Fig. 12 show that the systems worked best when they dealt with in-domain goals only and UP=5. In this best case, for Saplen we obtained that WA = 88%, CA = 84%, SU = 82%, IR = 2% and TC = 64%, whereas for Viajero we attained that WA = 78%, CA = 72%, SU = 71%, IR = 4% and TC = 69%. We observed in the generated dialogues two main reasons that caused speech recognition errors: acoustic similarity between some uttered words and strong accents of most speakers in our utterance corpora. Because of these errors, Saplen had problems in correctly understanding telephone numbers, street names, postcodes, building numbers and apartment letters, whereas Viajero had problems in correctly understanding telephone numbers, city names and day times. These findings indicate that we must improve the speech recognisers of the systems so that they can deal with these accents, for example, employing the lexicon or model adaptation methods discussed in [38–40].

As can be observed above, in the best case we obtained higher WA, CA and SU average scores for Saplen than for Viajero. However, the average TC score is higher for Viajero than for Saplen (69% vs. 64%)<sup>5</sup>. The reason for this surprising result can be explained from the scores set out in Table 5 and Table 7. Comparison of both tables shows that in the best case, the percentage of dialogues that ended without task completion because of excessive dialogue length is very high for Saplen (44%), whereas it is much lower for Viajero (7%). Analysis of the dialogues generated with Saplen in the best case reveals that the reason for such a high percentage is that the system employed too many confirmation prompts to obtain the user's post address, in accordance with its strategy for error handling explained in Section 4.4.1, thus making too many dialogues

to be cancelled by the user simulator. This finding indicates that we must improve this strategy to enhance the performance of the system's dialogue manager. To do so we could set up a strategy similar as the one employed by Viajero to deal with errors in bus travel bookings or queries, which requires less dialogue turns as discussed in Section 4.4.2.

Therefore, we can conclude that the proposed technique has been useful to exhaustively evaluating the systems, finding out problems regarding speech recognition, spoken language understanding, and dialogue management, which must be addressed to improve their performance. As said above, some of these problems are caused by the acoustic similarity between some uttered words and the strong accents of most speakers in our utterance corpora. Thus, we think these problems would have been difficult to uncover employing the user simulation techniques typically used nowadays, as they do not employ real speech and just consider the semantics of user intentions.

Future work includes making the proposed technique more easily applicable to other utterance corpora employed by the research community. In our experiments, we have applied the technique to deal with the corpora that we collected for previous studies, in which the information about the utterance type is included in the names of the files, for example, `telephoneNumber_20040312_19-52-39.wav`. However, in other utterance corpora the file names may be of the form: `utt00001.wav`, `utt00002.wav`, `utt00003`,..., which means that for these corpora our technique would not be directly applicable. The reason is that our algorithm to process the corpora model (see Fig. 3) would not be able to determine the utterance type from the file names. Hence, we plan to enhance our current implementation of the technique so that it can determine the utterance type from an inspection of the file that stores each semantic frame. For example, if the file `utt00001.RScor` contains the label `<telephoneNumber>` we may assume that the file `utt00001.wav` contains the utterance of a telephone number. To do so, we plan to set up a mapping between labels in semantic frames and utterance types, which should be easily reconfigured to ease the application of the proposed technique to different utterance corpora.

Moreover, it should be noted that to consider a broader range of different scenarios for system evaluation, our technique does not stick to a particular speaker when it creates the scenarios, and can employ utterances from different speakers. For example, it can create a scenario containing the goals `<telephoneNumber> = "958123456"` and `<postCode> = "18014"`, even though both goals are not uttered by the same speaker. This approach enables a large flexibility in the generation of simulated dialogues, which allows that the systems can be exhaustively evaluated. However, mixing speakers in one simulated dialogue is not a realistic situation and may have an impact on ASR. Thus, in future work we plan to enable speaker-sticking during the creation of scenarios and run new experiments to check the impact on the evaluation results.

## References

- [1] M.F. McTear, *Toward the conversational user interface*, Spoken Dialogue Technology, Springer, London, 2004.
- [2] S. Möller, *Quality of telephone-based spoken dialogue systems*, Springer, Berlin, US, 2004.
- [3] R. López-Cózar, M. Araki, *Spoken, Multilingual and Multimodal Dialogue Systems. Development and Assessment*, John Wiley & Sons Publishers, Chichester, 2005.
- [4] J. Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company Inc., Redwood, 1995.
- [5] K. Georgila, J. Henderson, O. Lemon, *Learning user simulations for information state update dialogue systems*, Proc. Interspeech (2005).
- [6] H. Ai, D.J. Litman, *Comparing real-real, simulated-simulated, and simulated-real spoken dialogue corpora*, in: Proceedings of AAAI workshop, 2006.

<sup>5</sup> These scores were obtained from the analysis of 5,026 generated dialogues for Saplen and 5,366 generated dialogues for Viajero.

- [7] E. Filisko, S. Seneff, Learning decision models in spoken dialogue systems via user simulation, in: Proceedings of AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems, 2006.
- [8] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, S. Young, Agenda-based user simulation for boot-strapping a POMDP dialogue system, Proc. NAACL-HLT (2007).
- [9] S. Jung, C. Lee, K. Kim, M. Jeong, D. Lee, G.G. Lee, Data-driven user simulation for automated evaluation of spoken dialogue systems, Comput. Speech Lang. 23 (2009) 479–509.
- [10] R. López-Cózar, A. De la Torre, J.C. Segura, A.J. Rubio, V. Sánchez, Assessment of dialogue systems by means of a new simulation technique, Speech Commun. 40 (3) (2003) 387–407.
- [11] R. López-Cózar, Z. Callejas, M. McTear, Testing the performance of spoken dialogue systems by means of an artificially simulated user, Artif. Intell. Rev. 26 (2006) 291–323.
- [12] S. Jung, C. Lee, K. Kim, D. Lee, G.G. Lee, Hybrid user intention modeling to diversify dialog simulations, Comput. Speech Lang. 25 (2011) 307–326.
- [13] H. Ai, D.J. Litman, Knowledge consistent user simulations for dialog systems, Proc. Interspeech (2007).
- [14] H. Ai, F. Weng, User simulation as testing for spoken dialogue systems, Proc. SIGDIAL (2008) 164–171.
- [15] F. Torres, E. Sanchis, E. Segarra, User simulation in a stochastic dialog system, Comput. Speech Lang. 22 (2008) 230–235.
- [16] D. Griol, Z. Callejas, R. López-Cózar, G. Riccardi, A domain-independent statistical methodology for dialog management in spoken dialog systems, Comput. Speech Lang. 28 (2014) 743–768.
- [17] S. Keizer, M.E. Foster, O. Lemon, A. Gaschler, M. Giuliani, Training and evaluation of an MDP model for social multi-user human-robot interaction, Proc. SIGDIAL (2013) 223–231.
- [18] E. Ferreira, F. Lefèvre, Reinforcement-learning based dialogue system for human-robot interactions with socially-inspired rewards, Comput. Speech Lang. 34 (1) (2015) 256–274.
- [19] J. Schatzmann, K. Weilhammer, M. Stuttle, S. Young, A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies, Knowl. Eng. Rev. 21 (2006) 97–126.
- [20] W. Eckert, E. Levin, R. Pieraccini, User modelling for spoken dialogue systems evaluation, Proc. ASRU (1997) 80–87.
- [21] E. Levin, R. Pieraccini, W. Eckert, A stochastic model of human-machine interaction for learning dialogue strategies, IEEE Trans. Speech Audio Process. 8 (1) (2000) 11–23.
- [22] K. Scheffler, S.J. Young, Corpus-based dialogue simulation for automatic strategy learning and evaluation, in: Proceedings of NAACL Workshop on Adaptation in Dialogue Systems, 2001, pp. 64–70.
- [23] O. Pietquin, A Framework For Unsupervised Learning of Dialogue Strategies Ph.D. Thesis, Faculte Polytechnique de Mons, 2004.
- [24] H. Ai, J. Tetreault, D. Litman, Comparing user simulation models for dialog strategy learning, Proc. NAACL-HLT (2007) 1–4.
- [25] G. Chung, Developing a flexible spoken dialog system using simulation, Proc. ACL (2004).
- [26] K. Georgila, J. Henderson, O. Lemon, User simulation for spoken dialogue systems: learning and evaluation, Proc. Interspeech (2006).
- [27] R. López-Cózar, Z. Callejas, D. Griol, Using knowledge of misunderstandings to increase the robustness of spoken dialogue systems, Knowl.-Based Syst. 23 (2010) 471–485.
- [28] R. López-Cózar, Using knowledge on word-islands to improve the performance of spoken dialogue systems, Knowl.-Based Syst. 88 (2015) 223–243.
- [29] R. López-Cózar, A.J. Rubio, P. García, J.C. Segura, Spoken dialogue system based on a dialogue corpus analysis, Proc. LREC (1998) 55–58.
- [30] R. López-Cózar, A.J. Rubio, J. Díaz Verdejo, A. De la Torre, Evaluation of a dialogue system based on a generic model that combines robust spoken language understanding and mixed-initiative control, Proc. LREC (2000) 743–748.
- [31] R. López-Cózar, D.H. Milone, A new technique based on augmented language models to improve the performance of spoken dialogue systems, Proc. Eurospeech (2001) 741–744.
- [32] T. Hain, P.C. Woodland, T.R. Niesler, E.W.D. Whittaker, The 1998 HTK system for transcription of conversational telephone speech, Proc. ICASSP (1999) 57–60.
- [33] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, P. Woodland, The HTK Book (for HTK Version 3.0), Cambridge, 2000.
- [34] A. Moreno, D. Poch, D. Bonafonte, E. Lleida, J. Llisterri, J.B. Mariño, C. Nadeu, ALBAYZÍN Speech database: design of the phonetic corpus, Proc. Eurospeech (1993) 175–178.
- [35] X. Huang, A. Acero, H. Hon, Spoken Language Processing: A Guide to Theory, Algorithm and System Development, Prentice-Hall, Upper Saddle River, 2001.
- [36] M. Boros, W. Eckert, F. Gallwitz, G. Götz, G. Hanrieder, H. Niemann, Towards understanding spontaneous speech: word accuracy vs. concept accuracy, in: Proceedings of 4th International Conference on Spoken Language Processing, 1996, pp. 1009–1012.
- [37] M. Danieli, E. Gerbino, Metrics for evaluating dialogue strategies in a spoken language system, in: Proceedings of AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation, 1995, pp. 34–39.
- [38] C. Huang, T. Chen, E. Chang, Accent issues in large vocabulary continuous speech recognition, Int. J. Speech Technol. 7 (2-3) (2004) 141–153.
- [39] U. Nallasamy, F. Metze, T. Schultz, Enhanced polyphone decision tree adaptation for accented-speech recognition, Proc. Interspeech (2012).
- [40] Y. Huang, D. Yu, C.J. Liu, Y.F. Gong, Multi-accent deep neural network acoustic model with accent-specific top layer using the KLD-regularized model adaptation, Proc. Interspeech (2014) 2977–2981.