Available online at www.sciencedirect.com

ScienceDirect

Speech Communication 50 (2008) 745–766

SPEECH COMMUNICATION

www.elsevier.com/locate/specom

ASR post-correction for spoken dialogue systems based on semantic, syntactic, lexical and contextual information

Ramón López-Cózar [*], Zoraida Callejas

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

Received 31 August 2007; received in revised form 9 January 2008; accepted 14 March 2008

Abstract

This paper proposes a technique to correct speech recognition errors in spoken dialogue systems that presents two main novel contributions. On the one hand, it considers several contexts where a speech recognition result can be corrected. A threshold learnt in the training is used to decide whether the correction must be carried out in the context associated with the current prompt type of a dialogue system, or in another context. On the other hand, the technique deals with the confidence scores of the words employed in the corrections. The correction is carried out at two levels: statistical and linguistic. At the first level the technique employs syntactic–semantic and lexical models, both contextual, to decide whether a recognition result is correct. According to this decision the recognition result may be changed. At the second level the technique employs basic linguistic knowledge to decide about the grammatical correctness of the outcome of the first level. According to this decision the outcome may be changed as well. Experimental results indicate that the technique enhances a dialogue system's word accuracy, speech understanding, implicit recovery and task completion rates by 8.5%, 16.54%, 4% and 44.17%, respectively.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Spoken dialogue systems; Speech recognition; Speech understanding; Natural language processing; Speech-based human–computer interaction

1. Introduction

As an effort to improve automatic information services making them available 24 hours a day, 365 days a year, many companies and official institutions have recently started to employ spoken dialogue systems (McTear, 2004; López-Cózar and Araki, 2005; Kraiss, 2006; Wahlster, 2006). These systems are computer programs developed to provide specific services using speech, for example, airplane travel information (Seneff and Polifroni, 2000), train travel information (Billi et al., 1997), weather forecasts (Zue et al., 2000; Nakano et al., 2001), fast food ordering (Seto et al., 1994; López-Cózar et al., 1997), call routing (Lee et al., 2000) or directory assistance (Kellner et al., 1997).

In despite of their advantages, spoken dialogue systems are rejected by many users because the interaction they allow is not very natural sometimes. This is caused by several reasons, but perhaps it is mainly a consequence of the current limitations of the state-of-the-art automatic speech recognition (ASR) for real-word applications (Rabiner and Juang, 1993; Huang et al., 2001; Skantze, 2005). Hence, to make dialogue systems more widely accepted, it is very important to develop techniques to increase the robustness of the speech recogniser employed by these systems. In order to make these techniques more easily employable by the research community, we believe that the techniques will require a small effort in corpus development, they must be easily applicable to different tasks, they will require small amounts of training data, and they must be easily implemented without too much programming cost.

Many studies can be found in the literature that aim to increase the robustness of a speech recogniser, for example, Levow (1998), Swerts et al. (2000), Mangu and

[*] Corresponding author. Tel.: +34 958 240579; fax: +34 958 243179.
E-mail addresses: rlopezc@ugr.es (R. López-Cózar), zoraida@ugr.es (Z. Callejas).

0167-6393/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.specom.2008.03.008

Padmanabhan (2001), Suhm et al. (2001), Levow (2002), Kakutani et al. (2002), Lo and Soong (2005), Ogata and Goto (2005), Shi and Zhou (2006), Denda et al. (2007) and Morales et al. (2007). ASR post-correction techniques aim to find incorrect words in the recognition result provided by the speech recogniser, and replace them with the correct ones, i.e. those uttered by the speaker. The techniques available so far in the literature, briefly discussed in Section 2, present several limitations. One is that many of them take into account lexical information only, and thus require large amounts of training data. Another drawback is that they not take into account the contextual information available in spoken dialogues. For example, if a spoken dialogue system prompts for a telephone number, it is likely that the user will utter specific kinds of words (digits) but not others (e.g. city names) to answer the prompt. Moreover, in a spoken dialogue the context in which words are uttered may change as the interaction proceeds. This kind of contextual information is missing in existing techniques, but according to our experiments, it can be very useful for the success of the ASR post-correction.

Another limitation of existing ASR post-correction techniques is that they consider the recogniser output just as a sequence of words, but do not take into account the confidence scores that may be attached to the words. However, many spoken dialogue systems employ these scores to decide whether to accept, reject or confirm (either implicitly or implicitly) the words in the speech recognition results (Hazen et al., 2002). The techniques available so far propose methods to replace a word w in a recognition result with another word w' in order to make a correction, but an open question is what should be the confidence score for the word w': should it be that of w or a different one? The technique we present in this paper addresses the drawbacks of the existing techniques discussed above.

The remainder of the paper is organised as follows. Section 2 presents previous work related to ASR post-correction, including differences and similarities with the proposed technique. Section 3 focuses on this technique. It firstly discusses the required elements: word-classes, grammatical rules, syntactic–semantic models and lexical models. Secondly, it explains algorithmically how to implement the technique. Thirdly, it analyses the performance of the technique in dealing with word insertions, substitutions and deletions, and then discusses the main advantages of the technique. Section 4 presents the experiments. Firstly, it shows the interaction between the Saplen system and the user simulator, and comments on the speech database and the scenarios for the simulator. Then it comments on experiments to decide the requirements on training data. Thirdly, the section compares results obtained with the baseline system and the proposed technique, and shows the advantages of using syntactic–semantic and lexical models, both contextual. Section 5 discusses limitations of the technique. Finally, Section 6 presents the conclusions and describes possibilities for future work.

2. Previous related work

Most previous studies on ASR post-correction are based on statistical methods that use probabilistic information about words uttered and words in the recognition results. For example, following this approach, Ringger and Allen (1996) proposed a post-processor to correct speech recognition errors based on two parts. A channel model represented errors made by a speech recogniser, whereas a language model represented the likelihood of a sequence of words uttered by the speaker. They trained both models with transcriptions of dialogues obtained with the TRAINS-95 dialogue system. Their experimental results showed that the post-processor output contained fewer errors than that of the speech recogniser. Also following this approach, Zhou and Meng (2004) proposed a two-level schema for detecting speech recognition errors. The first level applied an utterance classifier to decide whether the speech recognition result was erroneous. If it was determined to be incorrect, it was passed to the second level where a word classifier decided which words were misrecognitions.

Other methods employ co-occurrence information extracted from the words and their neighbouring words. For example, Zhou et al. (2006) proposed a method for error detection based on three steps. The first step was to detect whether the input utterance was correct. The second step was to detect incorrect words, and the third step was to detect erroneous characters. The error correction first created candidate lists of errors, and then re-ranked the candidates with a model that combines mutual information and a word trigram.

The methods discussed so far present several drawbacks. One is that they require large amounts of training data. Another is that their success depends on the size and quality of the speech recognition results or on the database of collected error strings, since they are directly dependent on the lexical entries. To address these drawbacks, researchers have employed additional knowledge sources. For example, Jeong et al. (1996) combined lexical information with semantic knowledge and carried out error correction at two levels: semantic and lexical. The input utterance was firstly transformed to obtain a lexico-semantic pattern. A database of pre-collected semantic patterns was used at the semantic level to find similar patterns to the obtained pattern. The error correction was made by replacing erroneous syntactic or semantic items in the obtained pattern, taking into account the pre-collected similar patterns. At the lexical correction level, the obtained and the recovered patterns were aligned, some candidate words in a domain dictionary or ontology dictionary were selected as the most similar to the original input words, and these words were used for correction.

Employing a different approach, Jeong et al. (2004) combined lexical information with higher level knowledge sources via a maximum entropy language model (MELM). Error correction was arranged on two levels, using a different language model at each level. At the first level, a word n-gram was employed to capture local dependencies and to speed up the processing. The MELM was used at the second level to capture long-distance dependencies and higher linguistic phenomena, and to re-score the N-best hypotheses produced by the first level. Their experiments showed that this approach had superior performance than previous lexical-oriented approaches. The problem was that the training of the MELM required a lot of time and was sometimes infeasible.

2.1. Differences and similarities between the proposed technique and previous studies

The technique we propose is inspired by previous studies based on semantic information (Jeong et al., 1996), pattern matching (Kaki et al., 1998) and statistical information (Zhou and Meng, 2004). One similarity between our technique and that of Jeong et al. (1996) is in the use of two correction levels. In both techniques the speech recognition result is transformed into one pattern. At the first level, this pattern is compared with a corpus of patterns learnt in the training, and as a result of the comparison the input pattern may be changed. The outcome of this level is passed on to the second level where both techniques can replace some words with other words.

One difference between both techniques is in the method employed to select the pattern to be used for making corrections at the first level. According to the technique of Jeong et al. (1996), this pattern is the one with minimum distance to the input pattern. One problem of this method is that the selected pattern may not be optimal. To overcome this problem, our method employs several corpora of previously learnt patterns, and a similarity threshold $t \in [0.0–1.0]$ to decide whether one pattern is good enough for error correction. If it is, our method works similarly to that of Jeong et al. (1996); otherwise it searches for a better pattern in the whole set of patterns available. If the appropriate pattern is found in the whole set, the correction proceeds as in the method of Jeong et al. (1996); otherwise, our technique does not make any correction at the first level. Another difference is that Jeong et al. (1996) carry out the lexical correction at the second level, whereas our method carries this correction both at the first and second levels. In the former level it considers statistical information, while in the latter it takes into account linguistic knowledge.

Our technique also has similarities with the proposal of Zhou and Meng (2004), as both employ a two-level schema for detecting recognition errors. The first level decides whether the speech recognition result is erroneous. If it is, the technique of Zhou and Meng (2004) passes on the result to the second level, where a word classifier decides which words are incorrect. One difference between the two techniques is that ours always passes on the result of the first level to the second level, regardless of the decision made by the first level.

Another difference between our technique and previous studies is that, as discussed in Section 1, existing studies focus just on the words in the speech recognition result, without considering the confidence scores that the words may have attached. Our technique not only deals with the word strings, but also with the confidence scores. As far as we know, this is an issue not addressed in previous studies.

3. The proposed technique

We propose a new ASR post-correction technique for spoken dialogue systems that can be useful when the speech recognition rate is very low. This is typically the case when such a system employs just one prompt-independent language model (e.g. word bigram) to recognise, in theory, any kind of sentence uttered by the user within the application domain, regardless of the current prompt generated by the system. Hence, our goal is to move forward from the typical solution employed by commercial systems, i.e. prompt-dependent language models mostly, to a more ambitious and less restricted interaction by means of just one prompt-independent language model, which enables the user to say anything within the application domain at anytime. We assume that the technique must be applicable to any dialogue system, regardless of the speech recogniser employed. Hence, we consider the recogniser as a black box that for each input utterance produces a recognition result. This result is a sequence of words with confidence scores[1] attached, e.g. "I (0.7590) would (0.6982) like (0.9268) to (0.4285) have (0.6929) six (0.3974) green (0.7059) salads (0.8182)". If the recogniser does not produce these scores but only the word sequence, our technique is applicable as well by simply discarding all the decisions about the scores.

The technique employs semantic, syntactic, lexical and contextual information. The semantic and syntactic information is implemented by means of sets of patterns constructed from the analysis of a dialogue corpus. Each set is mapped to a prompt type of a spoken dialogue system designed for a given application. The prompt type represents the contextual information of the technique, as it determines the kinds of sentence likely to be uttered by the user at a particular dialogue state. The lexical information is implemented by means of word confusion statistics and linguistic knowledge. The statistics are computed from the analysis of the dialogue corpus, by aligning utterance transcriptions and speech recognition results. The linguistic knowledge must be provided by the system designers and is employed to ensure grammatical restrictions (e.g. number correspondences) between specific words, in case these restrictions affect the semantics of the sentences. The reason for using this knowledge is to compensate for problems arising from sparse training data.

[1] We assume in this paper that the confidence scores are real numbers in the range [0.0–1.0].

3.1. Elements to be created for applying the technique

If we want to apply the technique to a spoken dialogue system, we must create the following elements: word-classes, grammatical rules, syntactic–semantic models and lexical models. The creation of these elements requires a corpus of system-user dialogues[2] that contains user utterances (voice samples files), transcriptions of the utterances (text sentences), and speech recognition results (text sentences) obtained by the speech recogniser of the dialogue system as it analyses the utterances.

We must assign a type (T) to each prompt the dialogue system can generate, taking into account the kind of data expected when the user answers the prompt. Note that a system may generate a prompt in different ways to avoid redundancy and increase the naturalness in the dialogue. For example, in the fast food domain a system can generate prompts such as "Would you like to have anything to eat?", "Anything to eat?" or "Are you ordering anything to eat?", but regardless of the wording, what is expected from the user is either a confirmation or a food order. Table 1 shows some assignments of types to prompts generated by the Saplen system used in the experiments.

3.1.1. Word-classes

The word-classes K_i are created using the set of utterance transcriptions in the dialogue corpus. Each word-class contains words of a given type that are really useful for the semantic analyser of the dialogue system to get the meaning of the sentences uttered by the user. These words are usually called *keywords*. The word-classes to be created can be decided by observing the transcriptions and using the system designers' knowledge about the performance of the semantic analyser. We call Ω the set of word-classes: $\Omega = \{K_1, K_2, K_3, \ldots, K_r\}$. Table 2 shows some possible word-classes in the fast food domain.

Usually a keyword belongs to just one word-class, e.g. 'want' belongs to the DESIRE class only. There may be keywords that belong to several word-classes, e.g. 'orange' could belong to the FOOD and TASTE classes. The technique requires the creation of word-classes for keywords only. Hence, word-classes are not required for meaningless words (e.g. articles or prepositions) if these words are not relevant for the semantic analyser of the dialogue system.

The word-classes are task-dependent. For example, in the ATIS (Air Travel Information Service) domain we could create the word-classes shown in Table 3.

The effort of determining the required word-classes for a given application may be relatively small using the designers' knowledge of the system, but filling the classes with all the required words can be costly if this is done manually, i.e. inserting the keywords one by one. However, if the semantic analyser of the dialogue system already uses word-classes to extract the meaning of the sentences uttered by the users, the technique can use these classes and there is no need to create them manually. This is the case for the Saplen system used in the experiments, where the semantic analyser uses word-classes stored in text files (one file per word-class). If the dialogue system does not use word-classes but uses finite-state grammars, which is the case for many commercial systems based on VoiceXML, it is possible to save time in creating the word-classes by re-using the vocabulary in the grammars, which is usually arranged in word categories. For example, Fig. 1 shows a JSGF grammar employed in a VoiceXML system that provides travel information. City names or week days can be easily copied and pasted to create the word-classes CITY and WEEK_DAY.

[2] This dialogue corpus can also be obtained by employing a user simulator that interacts with the dialogue system, as we have done in the experiments.

Table 1
Assignment of types to prompts generated by the Saplen system

Prompt	Type (T)
Please say your telephone number	TELEPHONE_NUMBER
I'm sorry, I didn't understand. Please say your telephone number again	TELEPHONE_NUMBER
What would you like to have?	PRODUCT_ORDER
How many ham sandwiches did you say?	FOOD_NUMBER
Which size would you like for the beer?	DRINK_SIZE
Did you say two?	NUMBER_CONFIRMATION
Please say the taste for the milkshake	DRINK_TASTE
Do you want to remove the green salad?	REMOVE_CONFIRMATION
Did you say large?	SIZE_CONFIRMATION

Table 2
Examples of word-classes in the fast food domain

Word-class	Word examples
CONFIRMATION	yes, no, OK, correct, incorrect, alright, etc.
DESIRE	want, need, gimme, etc.
DRINK	water, beer, coke, wine, fanta, etc.
FOOD	sandwich, cake, ice-cream, burger, etc.
INGREDIENT	cheese, ham, bacon, curry, etc.
NUMBER	one, two, three, four, five, six, etc.
SIZE	small, large, big, etc.
TASTE	orange, lemon, apple, etc.

Table 3
Examples of word-classes in the ATIS domain

Word-class	Word examples
DESIRE	want, need, etc.
CITY	Boston, Chicago, Rome, Madrid, Tokyo, etc.
WEEK_DAY	Monday, Tuesday, Wednesday, etc.
MONTH_DAY	first, second, third, fourth, etc.
MONTH_NAME	January, February, March, April, etc.
NUMBER	one, two, three, four, five, six, etc.
ACTION	book, cancel, confirm, question, etc.
DEPARTURE	from, departure, departs, etc.
ARRIVAL	arrive, arrives, arrival, etc.

```
#JSGF V1.0;

grammar from_to;

public <from_to> = [greeting] <desire> <travel> <city> {this.cityDestination=$city} <from_mark>
                          <city> {this.cityDeparture=$city} <moment> {this.weekDay=$moment} ;
<greeting> = hello | hi | good morning | good evening ;
<desire> = ( I | we ) | ( want | desire | need | must ) ;
<travel> = go to | travel to | get to ;
<city> = New York | Boston  | Chicago |  London | Madrid | Paris | Rome ;
<from_mark> = from | leaving from | departing from ;
<moment> = ( this | next ) ( Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday ) ;
```

Fig. 1. A sample JSGF grammar.

3.1.2. Grammatical rules

The grammatical rules are used to correct grammatical errors that affect the semantics of sentences. We call R the set of basic grammatical rules r_i: $R = \{r_1, r_2, \ldots, r_n\}$. The rules are of the form: r_i: $ssp_i \rightarrow restriction_i$, where ssp_i is a syntactic–semantic pattern and $restriction_i$ is a condition that must be satisfied by the words represented by the word-classes in ssp_i. For example, one rule used in our experiments in the fast food domain is the following:

$$r_1: \quad ssp_1 \rightarrow number(\text{NUMBER}) = number(\text{DRINK}) \textbf{ and } number(\text{DRINK}) = number(\text{SIZE}) \textbf{ and } number(\text{NUMBER}) = number(\text{SIZE})$$

where *number* is a function that returns either 'singular' or 'plural' for each word in the word-class, and $ssp_1 = \text{NUMBER DRINK SIZE}$.

One drawback of this method is that the rules are task-dependent. Hence, if the application domain is changed, the rules must be adapted accordingly to consider the necessary grammatical restrictions among the word-classes in the new domain. One alternative is to use statistical information, which has the advantage of automatic training without manual effort to create grammatical rules. To study the differences between both methods, we followed the study by Zhou et al. (2006) and used one *n*-gram language model, i.e. the word bigram employed by the speech recogniser of the Saplen system. We found that in some cases the statistical method worked well in correcting errors in speech recognition results, e.g. in "uno cervezas grandes" (one large beers), where 'uno' (one) was a misrecognition of 'dos' (two). The probability of this speech recognition can be expressed as follows:

$$P(\text{uno cervezas grandes}) = P(\text{uno}) \times P(\text{cervezas}|\text{uno}) \times P(\text{grandes}|\text{cervezas})$$

Using the bigram we found that the probabilities $P(\text{uno})$ and $P(\text{grandes}|\text{cervezas})$ were large as the word sequences 'uno' and 'cervezas grandes' were observed quite frequently in the training corpus. On the contrary, the probability $P(\text{cervezas}|\text{uno})$ was very small as the word sequence 'uno cervezas' was not observed. Thus we assumed that 'uno' was an incorrect word. We searched for an appropriate word in the trained lexical model (as will be explained in Section 3.1.4) and found the perfect candidate: 'dos' (two). Hence, we replaced 'uno' with 'dos' and had the misrecognition correctly corrected: "dos cervezas grandes" (two large beers).

The problem with this method is that because of sparse training data, there were cases where the method transformed correct recognition results into incorrect results, which is a known problem of for purely statistical methods as observed in previous studies (e.g. Kaki et al., 1998; Zhou et al., 2006). For example, in our application domain this happened with the recognition result "diez cervezas grandes" (10 large beers), which was not erroneous. For this result we found in the word bigram that the probabilities $P(\text{diez})$ and $P(\text{grandes}|\text{cervezas})$ were large, as the word sequences 'diez' and 'cervezas grandes' were observed, whereas the probability $P(\text{cervezas}|\text{diez})$ was very small as the word sequence 'diez cervezas' was not observed. Then we assumed, as in the above example, that 'diez' was an incorrect word. Hence, we searched for an appropriate substitution in the lexical model and found the candidate: 'tres'. Therefore, we replaced 'diez' with 'tres' and had the recognition result incorrectly corrected: "tres cervezas grandes" (three large beers).

Employing a large training corpus, we could have implemented this statistical method for making the correction at the lexical level, instead of the one based on grammatical rules. We decided to use the latter as it works better in cases of sparse data, given that the rules apply to all the words in the word-classes, regardless of whether the words have been observed in the training. Using the rule-based approach, the word 'diez' (10) in the example above would not have been considered an incorrect word as long as it is in the word-class NUMBER.

3.1.3. Syntactic–semantic models

The syntactic–semantic models are representations of the conceptual structure of the sentences in the application domain. To create these models we consider each prompt type T and take into account the transcriptions of all the sentences uttered by the users to answer system prompts of type T. For example, for the prompt type T = TELEPHONE_NUMBER we could find transcriptions such as: "My telephone number is nine five eight one two three four five six", "nine five eight one two three four five six" or "nine five eight twelve thirty-four fifty-six".

Each transcription must be transformed into what we call a *syntactic–semantic pattern* (*ssp*), which represents the sequence of word-class names in the transcription. We say that this kind of pattern is *syntactic* as it provides information about the order of the word-classes in the sequence. For example, the two *ssp*'s: ssp_1 = DESIRE NUMBER INGREDIENT and ssp_2 = NUMBER DESIRE INGREDIENT are syntactically different as they differ in the order of the word-classes. We say that this kind of pattern is also *semantic* as it provides information about the concepts (represented as word-classes) employed to convey some semantic content. For example, the two *ssp*'s: ssp_1 = DESIRE NUMBER FOOD and ssp_2 = DESIRE NUMBER DRINK are semantically different as they differ in the concepts involved (ssp_1 is a conceptual representation of a food order, whereas ssp_2 is a conceptual representation of a drink order).

To create a *ssp* from a transcription, each keyword in the transcription must be replaced with the name of the word-class the keyword belongs to. For example, taking into account the word-classes shown in Table 2 (Section 3.1.1), the *ssp* for the transcription: "I want one ham sandwich and one small beer please", would be as follows:

$$ssp = \text{DESIRE NUMBER INGREDIENT FOOD NUMBER SIZE DRINK}$$

If a keyword belongs to several word-classes, we include the names of the word-classes within brackets and separate them with the *optional* marker '|'. For example, if the keyword 'orange' belongs to the classes FOOD and TASTE, the *ssp* for the transcription "I want one orange Fanta" would be

$$ssp = \text{DESIRE NUMBER (FOOD|TASTE) DRINK}$$

Transforming all the utterance transcriptions associated with a prompt type T into *ssp*'s, we obtain a set of *ssp*'s associated with T. The next step is to analyse this set to remove repeated *ssp*'s, associating with each different *ssp* its relative frequency within the set. We call the result of this process a *syntactic–semantic model* associated with the prompt type T (SSM_T). This model is as follows:

$$\text{SSM}_\text{T} = \{ssp_1\ f_1, ssp_2\ f_2, ssp_3\ f_3, \ldots, ssp_m\ f_m\}$$

where the ssp_i's are syntactic–semantic patterns, and the f_i's are their relative frequencies within SSM_T. For example, the SSM_T associated with the prompt type T = TELEPHONE_NUMBER in our experiment is as follows:

$\text{SSM}_\text{TELEPHONE\_NUMBER}$ =
{NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER 0.1430,
NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER 0.1430,
NUMBER SIZE TASTE DRINK 0.0173,
MISTAKE 0.0098,
NUMBER INGREDIENT FOOD 0.0098,
...}

If a dialogue system generates *u* different prompt types T, we must create *u* different syntactic–semantic models SSM_T, one per prompt type. We call α set of all the SSM_T's for a given dialogue system:

$$\alpha = \{\text{SSM}_{\text{T}i}\}, \quad i = 1, \ldots, u$$

Note that a SSM_T is created from sentences uttered by users to answer a prompt type T. Hence, it is expected that the model contains syntactic–semantic patterns obtained from sentences actually related to the prompt type T, for example, patterns obtained from telephone numbers if the system prompted for a telephone number. However, a SSM_T can also contain *ssp*'s not directly related to the prompt type, which happens if the users uttered other types of sentence. For example, the $\text{SSM}_\text{TELEPHONE\_NUMBER}$ shown above contains a *ssp* obtained from food orders (NUMBER INGREDIENT FOOD 0.0098), another obtained from drink orders (NUMBER SIZE TASTE DRINK 0.0173) and another obtained from user corrections to system misunderstandings (MISTAKE 0.0098).

3.1.4. Lexical models

The lexical models contain information about the performance of the dialogue system's speech recogniser at each specific dialogue state. This state is represented by the prompt type T of the system. We must create a lexical model (LM_T) for each prompt type. The general form of this model is as follows:

$$\text{LM}_\text{T} = \{w_i \quad w_j \quad p_{ij}\}, \quad i,j = 1, \ldots, s$$

where w_i is a word uttered by the speaker, w_j is the recognition result, and p_{ij} is the posterior probability of obtaining w_j when the speaker uttered w_i. To create a LM_T we employ the set of utterance transcriptions and the set of recognition results associated with the prompt type T. Both sets are available from the analysis of the dialogue corpus. We align each transcription *c* with its corresponding recognition result *h* following the study of Fisher and Fiscus (1993), and add (w_i, w_j) pairs to LM_T, $w_i \in c$, $w_j \in h$. When all the transcriptions have been aligned, LM_T is analysed to remove repeated entries and to compute the probabilities p_{ij}. For example, let us suppose that the dialogue system prompted for the user's telephone number, i.e. T = TELEPHONE_NUMBER, and that to answer this prompt the user uttered the sentence: "nine five eight one two three four five six". Let us assume that the recognition result from this utterance is: "mine (0.3841) five (0.7867) eight (0.9345) one (0.7810) two (0.6721) three (0.8945) four (0.7832) five (0.7903) six (0.3981)". Then, following the procedure discussed above, the word pairs: ('nine', 'mine'), ('five', 'five'), ('eight', 'eight'), ('one', 'one'), ('two' , 'two'), ('three', 'three'), ('four', 'four'), ('five', 'five') and ('six', 'six') are added to LM_T. Hence, LM_T contains statistical information about the recognition of each word uttered by the user in response to the prompt type. After the removal of repeated entries and the computation of the word recognition probabilities, $LM_{TELEPHONE\_NUMBER}$ may be as follows:

$$\begin{aligned} LM_{TELEPHONE\_NUMBER} = \{ \; & \text{nine nine } 0.3679, \\ & \text{nine mine } 0.3530, \\ & \text{nine eight } 0.1457, \\ & \text{nine five } 0.1334, \\ & \text{six six } 0.8397, \\ & \text{six three } 0.1603, \\ & \text{salad salad } 0.7582, \\ & \text{salad salads } 0.2418, \\ & \ldots\} \end{aligned}$$

If a dialogue system generates u different prompt types T, we must create u different lexical models LM_T, one per prompt type. We call β the set of all the LM_T's for a given dialogue system:

$$\beta = \{LM_{Ti}\}, \quad i = 1, \ldots, u$$

As discussed for the syntactic–semantic models in the previous section, a LM_T is created from sentences uttered by users in the answering of system prompts of type T. Hence, it is expected that the model contains word recognition probabilities obtained from utterances actually related to the prompt type T, for example, recognition probabilities of digits if the system prompted for a telephone number. However, LM_T can also contain word recognition probabilities obtained from utterances not directly related to the prompt type, which happens if the users uttered other types of sentence. For example, the $LM_{TELEPHONE\_NUMBER}$ shown above contains recognition probabilities for the word 'salad', which in principle is not expected when the system prompted for the user's telephone number.

3.2. Algorithms to implement the technique

The ASR post-correction technique proposed in this paper is carried out firstly at the statistical level, and then at the linguistic level. The correction at the former level deals with the confidence scores if these scores are observed in the input utterance, otherwise the decision about these scores is skipped.

3.2.1. Correction at the statistical level

The goal of the correction at the statistical level is to take the result of the speech recogniser and employ statistical information to find words that belong to incorrect concepts, replace these concepts with correct concepts, and select the appropriate words for the correct concepts. If the algorithm finds that a word w belongs to an incorrect concept J and decides to replace it with another word w' that belongs to another concept K, it must decide the confidence score for the correction word w': $C(w')$. To do this we propose a simple method that takes into account the number of words $u_j \in K$ that are in the lexical model employed (either LM_T or β), i.e. words with which the word w is confused. These words form a set $U = \{u_1, u_2, u_3, \ldots, u_p\}$. The method to determine $C(w')$ is as follows: if there is just one word u_j in U, then $w' = u_j$ and $C(w') = 1.0$. If there are several words u_j's in U, then w' is the word with the highest confusion probability: p, and $C(w') = p$. This method assigns to w' the highest confidence score (1.0) if there is just one candidate word to make the word replacement, and assigns a smaller value if there are several candidates. We do not claim that this method is optimal, and in future work we will study other methods.

To implement the correction, the algorithm receives the speech recognition result and builds what we call an input *enriched syntactic–semantic pattern* ($essp_{INPUT}$), which is a sequence of information containers C_i as shown in Fig. 2.

Each container has an optional name which is the name of the word-class that contains the word w_i in the container. If this word is not in any word-class, the container has no name. If a container has a name, we say it is a *conceptual* container. The w_i's are the words in the speech recognition result, whereas the cs_i's are the confidence scores of the words. For example, using the word-classes shown in Table 2 (Section 3.1.1), the $essp_{INPUT}$ obtained from the recognition result: "I (0.5735) want (0.7387) one (0.6307) ham (0.3982) sandwich (0.6307) and (0.4530) one (0.6854) small (0.6590) beer (0.7861)" would be as shown in Fig. 3. Note that all the containers are conceptual, except the first and the sixth.

The algorithm now carries out two steps: pattern matching and pattern alignment, which are discussed below.

Step 1. Pattern matching: The goal of this step is to create an enriched syntactic–semantic pattern that we call $essp_{BEST}$. To do this we firstly define ssp_{INPUT} as the sequence of word-class names N_i in $essp_{INPUT}$. For example, ssp_{INPUT} for the $essp_{INPUT}$ shown in Fig. 3 would be

$$ssp_{INPUT} = \text{DESIRE NUMBER INGREDIENT FOOD NUMBER SIZE DRINK}$$

We decide whether ssp_{INPUT} matches any of the ssp's in the syntactic–semantic model associated with the prompt type T (SSM_T). If it does, we set $essp_{BEST} = essp_{INPUT}$ and proceed to the correction at the linguistic level. In this case

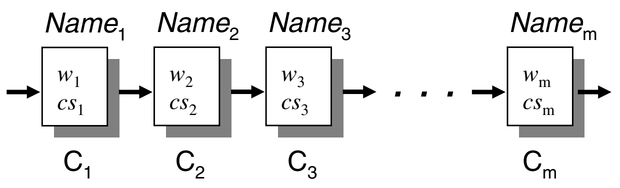

Fig. 2. General format of an input enriched syntactic–semantic pattern ($essp_{INPUT}$).

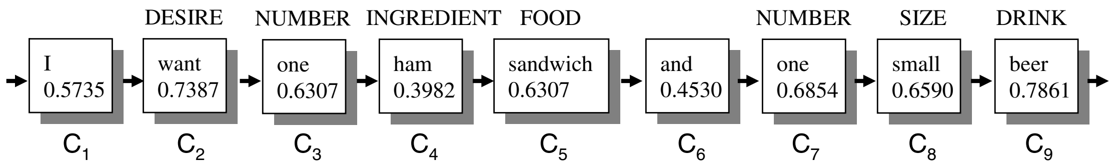

Fig. 3. Example of input enriched syntactic–semantic pattern ($essp_{\mathrm{INPUT}}$).

Step 2 is not necessary as no changes have been made in the sequence of concepts (i.e. word-class names) in ssp_{INPUT}. If ssp_{INPUT} does not match any pattern in $\mathrm{SSM_T}$, we search for similar patterns to ssp_{INPUT} in $\mathrm{SSM_T}$. To do this we compare ssp_{INPUT} with all the *ssp*'s in $\mathrm{SSM_T}$ employing the minimum edit distance dynamic search[3] (Crestani, 2000), and select those with *similarity* value greater than a threshold[4] $t \in [0.0\text{–}1.0]$. We compute the *similarity* of one pattern ssp_1 with respect to the other pattern ssp_2 as follows:

$$similarity(ssp_1, ssp_2) = (n - m_{\mathrm{ed}})/n$$

where n is the number of word-class names in ssp_1, and m_{ed} is the minimum edit distance between both patterns. Note that if ssp_1 and ssp_2 are exactly the same, their similarity is 1.0 given that $m_{\mathrm{ed}} = 0$. If they are completely different, their similarity is 0.0 because $m_{\mathrm{ed}} = n$. We call ssp_{SIMILAR} each *ssp* in $\mathrm{SSM_T}$ such that *similarity* $(ssp_{\mathrm{INPUT}}, ssp) > t$, and consider three cases:

Case 1. There is just one ssp_{SIMILAR} in $\mathrm{SSM_T}$. In this case the correction is made by setting $ssp_{\mathrm{BEST}} = ssp_{\mathrm{SIMILAR}}$ and proceeding to Step 2, which is now necessary as the sequence of concepts (i.e. word-class names) in ssp_{BEST} is different from that in ssp_{INPUT}.

Case 2. There are no ssp_{SIMILAR}'s in $\mathrm{SSM_T}$. This means that the *ssp* obtained from the recognition result is very different from all the *ssp*'s in $\mathrm{SSM_T}$. Then, we follow a fallback strategy and try to find ssp_{SIMILAR}'s in the α set (see Section 3.1.3). If no ssp_{SIMILAR}'s are found in α, this means that the obtained *ssp* is very different from all the *ssp*'s observed in the training, regardless of the system prompt type T. In this case we do not make any correction and proceed to the correction at the linguistic level. If just one ssp_{SIMILAR} is found in α, the correction is made as in Case 1, i.e. setting $ssp_{\mathrm{BEST}} = ssp_{\mathrm{SIMILAR}}$ and proceeding to Step 2. If several ssp_{SIMILAR}'s are found in α, we proceed as in Case 3.

Case 3. There are several ssp_{SIMILAR}'s in $\mathrm{SSM_T}$ or α. The question now is to decide the best ssp_{SIMILAR} to make the correction. To do this we start by selecting all the ssp_{SIMILAR}'s with the greatest *similarity* value. If there is just one, we set $ssp_{\mathrm{BEST}} = ssp_{\mathrm{SIMILAR}}$ and proceed to Step 2. If there are several, we select among them those with the highest relative frequency f_i in $\mathrm{SSM_T}$ or α. If there is just one, we set $ssp_{\mathrm{BEST}} = ssp_{\mathrm{SIMILAR}}$ and proceed to Step 2. If there are several, we do not make any correction at the statistical level and proceed to the correction at the linguistic level.

Step 2. Pattern alignment: Up to this point we haven taken ssp_{INPUT} and have created ssp_{BEST}. The former is of the form: $ssp_{\mathrm{INPUT}} = N_1 \quad N_2 \quad \cdots \quad N_m$, for example: $ssp_{\mathrm{INPUT}} =$ DESIRE INGREDIENT SIZE DRINK, whereas the latter is of the form: $ssp_{\mathrm{BEST}} = M_1 \quad M_2 \quad \cdots \quad M_r$, for example: $ssp_{\mathrm{BEST}} =$ DESIRE NUMBER SIZE DRINK. The goal of Step 2 is to create an enriched syntactic–semantic pattern that we call $essp_{\mathrm{BEST}}$, which is initially empty. We align ssp_{INPUT} and ssp_{BEST} and focussing on each container C_i in $essp_{\mathrm{INPUT}}$, we study two cases:

Case A. C_i is not conceptual. In this case the word w_i in C_i does not affect the semantics of the sentence, for example 'I' in container C_1 or 'and' in container C_6 of Fig. 3. Hence, we do not try to correct w_i. We simply set $D_i = C_i$ and add D_i to $essp_{\mathrm{BEST}}$ as observed in Fig. 4:

Case B. C_i is conceptual. In this case the word w_i in C_i affects the semantics of the sentence, for example 'one' in container C_3 of Fig. 3. Hence, we study whether this word must be changed considering the *ssp*'s observed in the training. Let us say that this container is N_a, $a \in 1, \ldots, m$, e.g. INGREDIENT in the ssp_{INPUT} shown above. We try to find the concept aligned with N_a in ssp_{BEST}. Let us say that it is M_b, $b \in 1, \ldots, r$, e.g. NUMBER in the sample ssp_{BEST} shown above, at the beginning of Step 2. We must consider three cases:

Case B.1. $N_a \neq M_b$. This is the case when a concept obtained from the speech recognition result (N_a) is considered to be incorrect and then must be replaced with another concept (M_b). We must decide the word $w'_i \in M_b$ and the confidence score sc'_i for the new container D_i to be added to $essp_{\mathrm{BEST}}$. To find this word we use the lexical model associated with the prompt type T ($\mathrm{LM_T}$) and create a set $U = \{u_1, u_2, u_3, \ldots, u_p\}$, $u_j \in M_b$, where the u_j's are words that are confused with

[3] The minimum edit distance between two syntactic–semantic patterns is defined as the number of deletions, insertions and substitutions of word-class names required for transforming one pattern into the other.

[4] The optimal value of the similarity threshold must be determined experimentally, employing the technique over a small set of the test database, and selecting as optimal the value that provides the best performance.

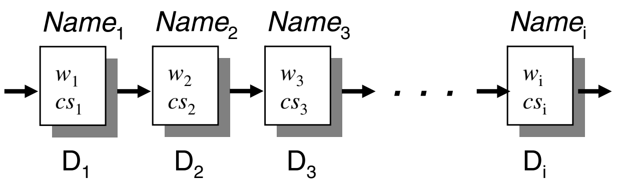

Fig. 4. $essp_{\mathrm{BEST}}$ with a new container D_i added.

the word w in N_a due to speech recognition errors. Three cases must be distinguished:

Case B.1.1. There is just one word u_j in U. Let us call this word u_1. We make[5]: $D_i.Name = M_b$, $D_i.w_i = u_1$, $D_i.cs_i = 1.0$, and add D_i to $essp_{\mathrm{BEST}}$.

Case B.1.2. U is empty. We follow a fallback strategy to find in the β set (see Section 3.1.4) the U set. If there is just one word u_j in U, we proceed as in Case B.1.1. If there are several words u_j's in U, we proceed as in Case B.1.3. If U is empty we do not make any correction, i.e. we make $D_i = C_i$ and add D_i to $essp_{\mathrm{BEST}}$.

Case B.1.3. There are several words u_j in U. We select the u_j with the highest confusion probability p, which we call it u_{high}. We make $D_i.Name = M_b$, $D_i.w_i = u_{\mathrm{high}}$ and $D_i.cs_i = p$, and add D_i to $essp_{\mathrm{BEST}}$.

Case B.2. $N_a = M_b$. This is the case when the concept obtained from the speech recognition result is assumed to be correct. Hence, we do not change it. We simply make $D_i = C_i$ and add D_i to $essp_{\mathrm{BEST}}$.

Case B.3. N_a cannot be aligned. This is the case when the concept obtained from the speech recognition result is a consequence of an inserted word. To correct the error we discard C_i, i.e. we do not add it to $essp_{\mathrm{BEST}}$.

3.2.2. Correction at the linguistic level

Up to this point we have created $essp_{\mathrm{BEST}}$. The correction now aims to correct errors not detected at the statistical level that affect the semantics of the sentences. For example, we have observed in the experiments that when the system prompts to enter product orders, the utterance "una cerveza grande" (one large beer) is sometimes recognised as "dos cerveza grande" (two large beer), which causes the system to incorrectly understand the order as "two large beers". This kind of problem is not detected at the statistical level as the *ssp* obtained from the incorrect recognition result:

$ssp_{\mathrm{INPUT}} = $ NUMBER SIZE DRINK

matches one of the *ssp*'s in $\mathrm{SSM_T}$. Hence, ssp_{INPUT} is not corrected at the statistical level, which results in the $essp_{\mathrm{BEST}}$ *obtained so far being incorrect. To solve this problem we use the set of grammatical rules R* discussed in Section 3.1.2. We place in a window each syntactic–semantic pattern ssp_i in a rule r_i. The window slides over $essp_{\mathrm{BEST}}$ from left to right. For example, Fig. 5 shows a sample window for the rule r_1 discussed in Section 3.1.2.

If the sequence of concepts in the window is found in $essp_{\mathrm{BEST}}$, $restriction_i$ applies to the words in the containers. For example, in Fig. 5 the sequence of concepts is found in the sequence of containers C_7–C_9. Therefore, $restriction_1$ applies to the words: 'two', 'small' and 'beer'. If the conditions in $restriction_1$ are satisfied we do not make any correction, otherwise we try to find the reason for the incongruity by searching for an incorrect word. This is the case in the example given that $number(\mathrm{NUMBER}) \neq number(\mathrm{DRINK})$. To find an incorrect word we use the linguistic information available by means of the grammatical rules. In our experiments this is information about the number feature of some Spanish words, for example: *number*('dos') = plural, *number*('cerveza') = singular, and *number*('pequeña') = singular (dos = two, cerveza = beer, pequeña = small). By comparing the number features of these words, we assume that the word 'dos' (two) in container C_7 is incorrect, as the number correspondence between 'dos' (two) and 'cerveza' (beer) is incorrect. Hence we define $IncorrectContainer = C_7$ and proceed similarly as explained in Section 3.2.1, Step 2, Case B.1. However, now the goal is not to replace a concept with another concept, but a word within a concept with another word within the same concept. To do this replacement we use the lexical model $\mathrm{LM_T}$ and define the set $U = \{u_1, u_2, u_3, \ldots, u_p\}$, where all the words u_j's belong to the same word-class as the word to be replaced (i.e. NUMBER in the example), are confused with this word and satisfy $restriction_i$. We finally consider the three cases distinguished in Section 3.2.1, Step 2, Case B.1, and proceed as discussed there.

3.2.3. Analysis of the performance of the technique for word insertions, substitutions and deletions

Regarding word insertions, the correction is successful if the technique discards the concepts that appear in ssp_{INPUT} due to inserted words if these words are keywords. The correction can be observed in the following example taken from our experiments, where T is the prompt type of the dialogue system, U is the sentence uttered by the user, h is the speech recognition result, and h' is the result of the ASRPC module that implements the proposed technique (see Fig. 6):

T = FOOD_ORDER_CONFIRMATION
U = one curry salad
h = one (0.8982) **error** (0.6950) curry (0.5982) salad (0.8059)
h' = one (0.8982) curry (0.5982) salad (0.8059)

The inserted word 'error' caused the concept ERROR to be in the obtained syntactic–semantic pattern: $ssp_{\mathrm{INPUT}} =$ NUMBER ERROR INGREDIENT FOOD. The technique selects the learnt pattern: NUMBER INGREDIENT FOOD as the most similar pattern to ssp_{INPUT}, and uses it for correction, discarding the concept ERROR at the pattern alignment step.

Regarding word substitutions, the correction requires on the one hand that the technique correctly replaces the incorrect concepts that appear in ssp_{INPUT} because of the substituted words (assuming that the words are keywords). The replacement can be observed in the following example:

T = TELEPHONE_NUMBER
U = **nine** one two three four five six seven eight

[5] We use the notation '$D_i.Name$', '$D_i.w_i$' and '$D_i.cs_i$' to refer to the fields *Name*, w_i and cs_i of the container D_i, respectively.

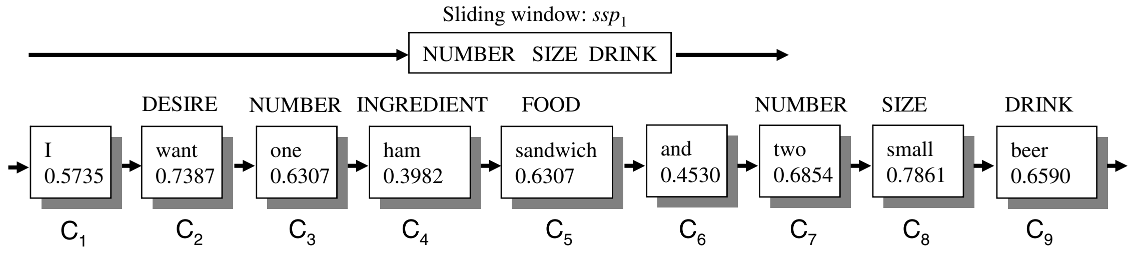

Fig. 5. Window sliding over $essp_{\text{BEST}}$.

h = **gimme** (0.4982) one (0.6950) two (0.5982) three (0.6059) four (0.8691) five (0.6892) six (0.5723) seven (0.5211) eight (0.8561)
h' = nine (1.0000) one (0.6950) two (0.5982) three (0.6059) four (0.8691) five (0.6892) six (0.5723) seven (0.5211) eight (0.8561)

We can see that the uttered word 'nine' was substituted by the word 'gimme'. Hence, ssp_{INPUT} = DESIRE NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER. The technique selects the learnt pattern: NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER as the most similar learnt pattern to ssp_{INPUT}. At the pattern matching step it replaces the concept DESIRE with the concept NUMBER, thus correcting the incorrect concept. On the other hand, the correction requires that the technique finds the appropriate candidate word to replace the incorrect word. This is the case in the example, as the technique searches in the word-class NUMBER for a candidate for the word 'gimme' and finds the word 'nine', which is the word uttered by the user.

If the error substitutes a word with another word and both words are keywords in the same word-class, there is no conceptual correction. This is the case, for example, if the user utters the sentence "two ham sandwiches" and it is recognised as "one ham sandwiches", where 'two' and 'one' are keywords in the word-class NUMBER. In this case the correction is successful only if it is successful at the linguistic level.

There is no conceptual correction if the error substitutes a non-keyword with another non-keyword. This happens, for example, if the utterance "please two ham sandwiches and two beers" is recognised as "uhm two ham sandwiches uhm two beers". The conceptual correction fails as there is no change in the sequence of concepts obtained from the utterance.

The technique cannot correct word deletion errors. By carrying out a comparison with the learnt patterns, it can detect that one or more concepts are missing in ssp_{INPUT} because of the deletion, but it cannot decide the words to fill in the gaps because there are no words that can be used as candidates for the search (given that these words have been deleted). This problem can be observed in the following example:

T = TELEPHONE_NUMBER
U = nine five eight three **two** zero three one seven
h = nine (0.6450) five (0.7941) eight (0.6019) three (0.4002) zero (0.4735) three (0.8998) one (0.8647) seven (0.6953)
h' = nine (0.6450) five (0.7941) eight (0.6019) three (0.4002) zero (0.4735) three (0.8998) one (0.8647) seven (0.6953)

We can see that the uttered word 'two' was deleted. Hence, ssp_{INPUT} = NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER. The technique selects the pattern: NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER as the most similar learnt pattern to ssp_{INPUT}. At the pattern matching step, the technique aligns all the concepts in ssp_{INPUT} with those in the most similar pattern. The missing concept in ssp_{INPUT} because of the deleted word has no effect on the matching, and thus there is no correction.

3.2.4. Advantages of the technique

The technique requires only a small effort in corpus development if the dialogue corpus needed for training can be easily collected. This is the case when the dialogue system is running in a commercial application or in a research environment, as in these settings it is usually possible to automatically collect a corpus. If the system is not running in any of these settings, we must collect the corpus from the start, or use a corpus that is already available for the same application domain, perhaps collected using another dialogue system. Hence, this satisfies only partly one of our initial goals when developing the technique, i.e. minimal initial effort in corpus development.

The effort for assigning types (T) to the prompts generated by a dialogue system is very small, which satisfies one of our initial goals when developing the technique: easy application to different tasks. Using grammatical rules instead of statistical information for the correction at the linguistic level, satisfies another of our initial goals: minimal requirements for training data. The simple algorithms discussed in Section 3.2 make it easy to set up the technique, which satisfies another initial goal: easy implementation.

The technique learns very rapidly from the training data. This happens because the syntactic–semantic structure of the sentences used for training is represented by means of patterns comprised of word-classes. This kind of pattern allows us to generalise knowledge obtained from the training, and to apply it to cases not observed in the training (Ward and Issar, 1996).

The technique is robust against some spontaneous speech phenomena. For example, it can handle hesitations (e.g. 'uhm') typically uttered by users when they are thinking what to say next in the dialogue. This happens because hesitations are not keywords, and thus they do not affect the sequence of concepts obtained from the analysis of the sentence. For example, if the user utters either "uhm…one ham sandwich" or "one ham sandwich", ssp_{INPUT} is: NUMBER INGREDIENT FOOD.

The technique is also robust against repeated words typically uttered in spontaneous speech, provided that this phenomenon is observed in the training dialogue corpus. For example, if the sentence "one…uhm…one…ham sandwich" is in this corpus, the technique learns the syntactic–semantic pattern: $ssp =$ NUMBER NUMBER INGREDIENT FOOD. In this way, if a user utters the sentence "two…uhm…two vegetarian salads please", ssp_{INPUT} is: NUMBER NUMBER INGREDIENT FOOD, and thus the technique does not make any correction at the pattern matching step. Note that this pattern is also useful for the *changes of mind* typical of spontaneous speech, where the user corrects data as he speaks. This phenomenon can be observed in the following example: "one…uhm…well…uhm…two ham sandwiches please". For this sentence, ssp_{INPUT} is also: NUMBER NUMBER INGREDIENT FOOD, and thus the technique does not make any correction at the pattern matching step.

The effect of partially uttered words, also typical of spontaneous speech, depends on the kind of speech recognition error they cause. If by chance these words cause a kind of error observed in the training, the error may be corrected. For example, let us suppose that the user utters the sentence "I would like one sm…small beer please", where he partially utters the word 'small'. If the recognition result is e.g. "I would like one is small beer please" the error can be corrected as ssp_{INPUT} would be: DESIRE NUMBER SIZE DRINK, which is observed in the training. The technique would discard the word 'is' as it is not a keyword. The technique would also be successful if the recognition result was: "I would like one s small beer please". For this input ssp_{INPUT} would be: DESIRE NUMBER LETTER SIZE DRINK, given that the word 's' is the LETTER word-class. As the most similar learnt pattern to the input pattern would be: DESIRE NUMBER SIZE DRINK, the result of the pattern alignment would be the removal of the LETTER concept, and thus the error would be corrected as the word 's' would be discarded.

It is possible to think of cases where the technique would fail when dealing with partially uttered words. For example, the user could utter the sentence "one ve…uhm one curry salad please" where he changes his mind, leaving the word 'vegetarian' partially uttered. The recognition result for this utterance could be e.g. "one beer and one curry salad please" and thus ssp_{INPUT} would be: NUMBER DRINK NUMBER INGREDIENT FOOD. As this pattern is observed in the training, there would be no correction at the pattern matching step and the error would remain uncorrected.

4. Experiments

The goal of the experiments was to test the effect of the proposed technique on the performance of the Saplen system. To do this we compared evaluation results obtained with two speech recognition front-ends:

(i) The standard HTK-based speech recogniser of the Saplen system (baseline).
(ii) The enhanced speech recogniser shown in Fig. 6, which employs the ASRPC (ASR Post-Correction) module that implements the proposed technique.

In the figure T denotes the current prompt type of the dialogue system, Ω represents the set of word-classes, R is the set of grammatical rules, α is the set of syntactic–semantic models, β is the set of lexical models, and t is the similarity threshold. The input to the ASRPC module was a recognition result provided by the speech recogniser: $h = w_1 cs_1 w_2 cs_2 \cdots w_n cs_n$, where the w_i's represent words and the cs_i's confidence scores. The output of the module was another speech recognition result: $h' = w'_1 cs'_1 w'_2 cs'_2 \cdots w'_m cs'_m$, where some words and confidence scores may be changed.

The performance of the HTK-based recogniser and the ASRPC module was saved in log files for evaluation purposes. Each entry in these files contained the transcription of each utterance, the speech recognition result (h), the result of the ASRPC module[6] (h'), and a tag indicating whether the speech recognition result was correctly understood by the Saplen system.

4.1. Evaluation measures

Experimental results were obtained in terms of word accuracy (WA), speech understanding (SU), implicit recovery (IR) and task completion (TC) (Danieli and Gerbino, 1995). WA is the proportion of correctly recognised words. It was computed as the percentage $\mathrm{WA} = (w_t - w_i - w_s - w_d) \times 100/w_t$, where w_t is the total number of words in the analysed sentences, and w_i, w_s and w_d are the numbers of words inserted, substituted and deleted by the speech recogniser of the Saplen system, respectively.

Sentence understanding (SU) is the proportion of sentences correctly understood by the Saplen system as it interacted with the user simulator (see Fig. 7). It was computed as the percentage $\mathrm{SU} = S_u \times 100/S_t$, where S_u is the

[6] This data was stored only when the ASRPC module was used.

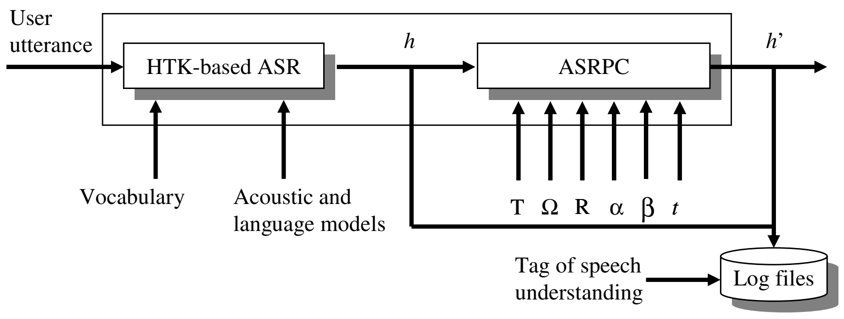

Fig. 6. Enhanced speech recogniser, including the ASRPC technique.

number of analysed sentences where the semantic representation obtained by the system was completely correct (i.e. it was exactly the same as the correct semantic representation) and S_t is the total number of sentences analysed.

The semantic analyser of the Saplen system is robust against some kinds of recognition error, which enabled it to correctly understand some sentences even though some words were misrecognised. Implicit recovery (IR) is the proportion of incorrectly recognised sentences that were correctly understood by the system. It was computed as the percentage $\mathrm{IR} = S_u \times 100/S_e$, where S_u is the number of analysed sentences in which the obtained semantic representation was completely correct, and S_e is the number of incorrectly recognised sentences.

Task completion (TC) is the proportion of successful dialogues, i.e. the percentage of dialogues that ended with the achivement of all the goals in the scenario used for each dialogue simulation (see Fig. 8). It was computed as the percentage $\mathrm{TC} = D_c \times 100/D_t$, where D_c is the number of successful dialogues and D_t is the total number of dialogues. As will be discussed in the following section, the user simulator cancelled the interaction with the dialogue system if the dialogue became too long due to system malfunction. Cancelled dialogues were not considered successful and thus decreased the TC rate.

4.2. Interaction between the Saplen system and the user simulator

The Saplen system was developed in a previous study to answer Spanish telephone-based orders and queries by clients of fast food restaurants (López-Cózar et al., 1997). To develop the system we collected a dialogue corpus in a fast food restaurant that contains about 800 recorded dialogues in Spanish involving conversations between clients and restaurant assistants (López-Cózar et al., 1998). These dialogues contain product orders, telephone numbers, postal codes, addresses, queries, confirmations, greetings and other types of sentence. The dialogues were transcribed, labelled and analysed to include tags regarding the speakers (clients and restaurant assistants), sentence types, semantic information of sentences and other kinds of information. From this corpus we created the Saplen corpus which contains 5500 client utterances and about 2000 different words. It also contains the utterance transcriptions and the semantic representations of the utterances stored as *frames* (Allen, 1995). We have used this corpus for previous studies (e.g. López-Cózar et al., 2003; López-Cózar and Callejas, 2005).

In a previous study we developed a user simulator the purpose of which is to interact automatically with a dialogue system to obtain a dialogue corpus suitable for testing purposes (López-Cózar et al., 2003). The most recent version of the simulator supports three different types of user: very cooperative, cooperative and not very cooperative, in order to simulate a wider variety of user inputs (López-Cózar et al., 2006). When simulating a very cooperative user, the simulator always provides the type of data the system asks for, and when the system prompts to confirm data, it always answers an appropriate affirmative or a negative confirmation. When simulating a cooperative user, the simulator always provides the type of data the system asks for, and when the system prompts to confirm data, it sometimes answers an appropriate affirmative or a negative confirmation, and in other cases it repeats the data being confirmed. When simulating a not very cooperative user, the simulator selects randomly the type of data it provides when the dialogue system asks for a particular data, and when the system prompts to confirm data it behaves as a cooperative user. Fig. 7 shows the connection between the user simulator and the Saplen system (the speech synthesiser is not shown as it is not used in these experiments).

The user simulator receives the current prompt generated by the dialogue system as well as the semantic representation obtained by the system from the analysis of the previous simulator's response. Therefore, the semantic representation can be affected by recognition and understanding errors similarly as if the simulator response had been uttered by a real user. To interact with the dialogue system the user simulator employs a set of scenarios that indicate the goals it must try to achieve during the interaction. For example, the scenario shown in Fig. 8 indicates that the simulator must order initially “one large cola” and “one ham sandwich”. When the system prompts for additions to the order, it must order “one green salad”. The scenario goals are semantic representations (frames) of utterances in the corpus.

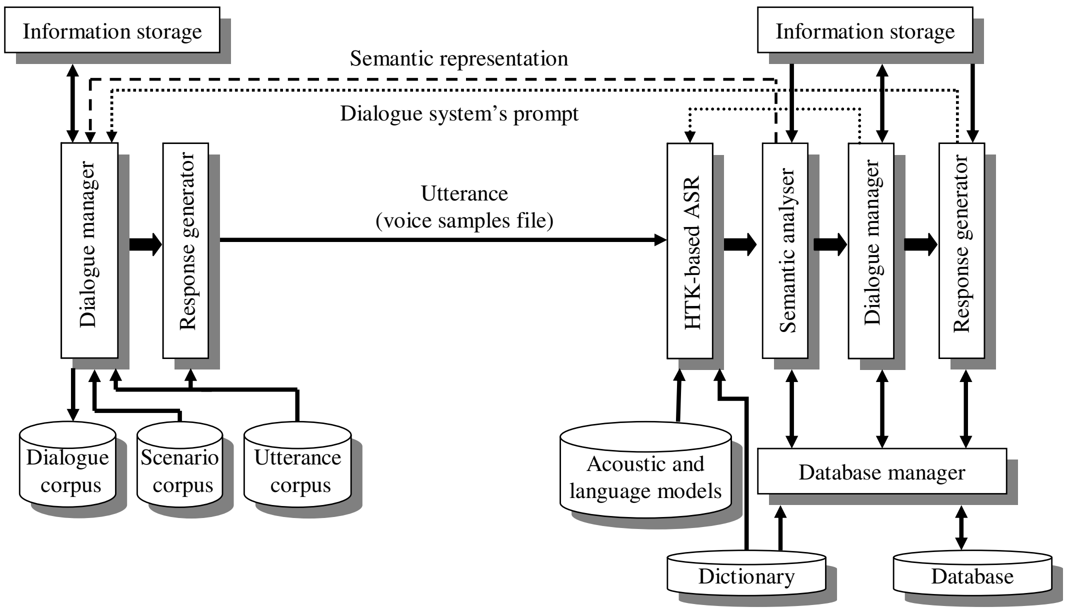

Fig. 7. The user simulator (on the left) interacting with the Saplen system (on the right).

# the user wants to order one large cola and one ham sandwich [<AMOUNT> = ”1”, <DRINK> = ”COLA”, <SIZE> = ”LARGE” <AMOUNT> = ”1”, <FOOD> = ”SANDWICH”, <INGREDIENTS> = ”HAM”] # the user wants to order a green salad [<AMOUNT> = ”1”, <FOOD> = ”SALAD”, <INGREDIENTS> = ”GREEN”]	# the user telephone number [<TELEPHONE_NUMBER> = ”958122345”] # the user postal code [<POSTAL_CODE> = ”18001”] # the user address [<ADDR_TYPE> = ”STREET”, <ADDR_NAME> = ”ACERA DE CANASTEROS”, <ADDR_NUMBER> = ”1”, <ADDR_FLOOR> = ”THIRD”, <ADDR_LETTER> = ”E”]

Fig. 8. A sample scenario for the user simulator.

When the system prompts for the user's telephone number, postal code or address, the user simulator employs the telephone number, postal code and address in the scenario. When the appropriate scenario goal, e.g. <POSTAL_CODE> = “18001”, is selected to answer a system prompt, the simulator selects at random[7] any utterance in the corpus (voice samples file) the semantic representation of which matches the selected goal, for example “my postal code is 18001”. This utterance is the input for the dialogue system's HTK-based speech recogniser, as can be observed in Fig. 7. The dialogues generated in the system–simulator interaction are stored in log files for evaluation purposes, including the system prompt, the prompt type (T), the user simulator's response (utterance transcription) and the speech recognition result. The simulator employs an *interactionLimit* parameter to cancel the dialogue with the dialogue system if the total number of interactions (i.e. of system plus user simulator) exceeds a threshold set to 30 interactions. The value of this parameter was decided considering the complexity of the scenarios employed for the interaction. Below is a sample dialogue (translated from Spanish into English) between the system and the user simulator behaving as a not very cooperative user.

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

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

T = ANYTHING_TO_ORDER

(2) *User simulator*: Acera de Canasteros Street, number one, third floor, letter e [Recognition result: cherry (1.0000) make (0.5997) an (0.0718) order (0.718) of (0.6070) elm (0.4084)].
(3) *Saplen*: I'm sorry, I didn't understand. You can order products or ask for information. How may I help you?

T = ORDER_OR_INFORMATION

(4) *User simulator*: one large cola and one ham sandwich [Recognition result: one (0.7991) large (0.9785) cola (0.4997) and (0.4871) one (0.4942) ham (0.6591) sandwich (0.6781)].
(5) *Saplen*: OK. One large cola and one ham sandwich. Would you like anything else?

T = ANYTHING_TO_ORDER

(6) *User simulator*: one green salad [Recognition result: one (0.8564) green (0.8564) salad (0.5864)].
(7) *Saplen*: OK. Please say your telephone number.

T = TELEPHONE_NUMBER

(8) *User simulator*: 9 5 8 12 23 45 [Recognition result: eighteen (1.0000) zero (0.8354) zero (0.6923) seven (0.4999) forty-five (0.3870)].
(9) *Saplen*: I need to confirm your telephone number. Did you say 18, 0, 0, 7, 45?

T = TELEPHONE_CONFIRMATION

(10) *User simulator*: 9 5 8 12 23 45 [Recognition result: seventy-eight (0.5025) twelve (0.7002) seventy-five (0.6698)].
(11) *Saplen*: OK. Telephone number 78, 12, 75. Please say your postal code

T = POSTAL_CODE

(12) *User simulator*: This is incorrect [Recognition result: this (0.4048) is (0.6992) incorrect (1.0000)].
(13) *Saplen*: OK. Please say your telephone number again.

T = TELEPHONE_NUMBER

(14) *User simulator*: 9 5 8 12 23 45 [Recognition result: nine (0.9390) five (0.9581) eight (0.3919) twelve (0.6750) twenty-three (0.7848) forty-five (0.9816)].
(15) *Saplen*: I must confirm your telephone number again. Did you say 9, 5, 8, 12, 23, 45?

T = TELEPHONE_CONFIRMATION

(16) *User simulator*: 9 5 8 12 23 45 [Recognition result: nine (0.7835) five (0.8273) eight (0.8471) twelve (0.9849) twenty-three (0.6084) forty-five (0.8481)].
(17) *Saplen*: OK. Telephone number 9, 5, 8, 12, 23, 45. Please say your postal code

T = POSTAL_CODE

(18) *User simulator*: 18 0 0 1 [Recognition result: eighteen (0.7984) zero (1.0000) zero (1.0000) one (0.5048)].
(19) *Saplen*: OK. Postal code number 18, 0, 0, 1. Please say your address, for example, Elm street, number 13, first floor, letter e

T = ADDRESS

(20) *User simulator*: Acera de Canasteros street, number one, third floor, letter e [Recognition result: acera-de-canasteros (1.0000) street (1.000) ice (0.3046) error (0.3046) twelve (1.0000) beer (0.1862)].
(21) *Saplen*: I have to confirm the building number. I think you said 12. Is this correct?

T = BUILDING_NUMBER

(22) *User simulator*: one [Recognition result: two (0.4223)].
(23) *Saplen*: I have to confirm the building number again. Did you say 2?

T = BUILDING_NUMBER_CONFIRMATION

(24) *User simulator*: one [Recognition result: one (0.5891)].
(25) *Saplen*: OK. Number one. Please say the floor of the apartment

T = APARTMENT_FLOOR

(26) *User simulator*: third [Recognition result: first (0.8012)].
(27) *Saplen*: OK. First floor. Please say the apartment letter

T = APARTMENT_LETTER

(28) *User simulator*: This is incorrect [Recognition result: this (0.5091) is (0.9892) incorrect (0.7432].
(29) *Saplen*: OK. Please say the floor again

T = APARTMENT_FLOOR

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

Employing this kind of dialogue we aim to simulate real dialogues in which the users sometimes utter sentences that do not match the current prompt of the dialogue system, as can be observed in the following interaction:

(1) *Human operator*: Good evening, welcome to our restaurant. What would you like to have?
(2) *User*: one ham sandwich and one large beer please.
(3) *Human operator*: Anything else?
(4) *User*: no, thanks.
(5) *Human operator*: What is your telephone number?
(6) *User*: well, uhm... I think I'd better have one small beer instead of one large beer.
(7) *Human operator*: OK, small beer. Telephone number, please?

Turn (6) of this example would not be recognised if at this dialogue state the system used a prompt-dependent language model trained from telephone numbers only. Hence, it is important to use wide coverage grammars that allow the system, in principle, to recognise any kind of sentence permitted in the application domain, regardless of the current prompt of the dialogue system. The problem is that these grammars are more complex, and the vocabulary active at each dialogue state is much larger, which tends to markedly increase speech recognition errors.

4.3. The speech database and test scenarios for the user simulator

We used two separate utterance corpora, one for training and the other for testing, that we employed in a previous study (López-Cózar et al., 2003). Both disjunct corpora were created using the Saplen corpus discussed in the previous section, selecting utterances at random among the 18 types shown in Table 4. Each corpus included the transcriptions of the utterances as well as their corresponding semantic representations.

We compiled one prompt-independent language model (word bigram) from the 2750 training sentences to enable the recognition of the 18 sentence types set out in Table 4. In theory, this language model would provide users with a more natural interaction as they could utter any of the 18 sentence types at any moment in the dialogue, regardless of the current prompt generated by the system. However, we observed in previous experiments that using such a grammar degraded the word accuracy given that the vocabulary was large and there were many possible sentences to be considered during the analysis of each utterance. This problem motivated the proposed technique, which is intended to increase word accuracy when such a language model is used.

Employing the test utterance corpus we automatically created 400 different scenarios similar to that shown in Fig. 8. To do this we used the semantic representations of the 250 product orders, telephone numbers, postal codes and addresses in this corpus. Each scenario contained one product order, which was created by means of the random combination of 1–3 semantic representations of product orders. The scenario also contained the semantic representation of one telephone number, one postal code and one address, also selected at random. To carry out the experiments the scenario corpus was divided into two separate scenario corpora, which we called *ScenariosA* (300 scenarios) and *ScenariosB* (100 scenarios).

4.4. Experiments to decide the requirements on training corpus

We initially carried out additional experiments to decide the appropriate number of dialogues that enabled the syntactic–semantic and lexical models to obtain the maximum amount of information from the training. We employed the Saplen system and the user simulator and increased the number of automatically generated dialogues until we did not observe any change in the amount of learnt information, nor in terms of syntactic–semantic patterns in the α set, nor in terms of lexical confusions in the β set. Table 5 shows the results obtained.

It can be observed that the number of syntactic–semantic patterns and lexical confusions increased with the number of generated dialogues until it reached a threshold. The maximum number of patterns (175) was obtained for 750 generated dialogues, whereas the highest number of lexical confusions (1042) was achieved for 900 dialogues. Hence, we assumed that 900 was the optimum number of dialogues that needed to be generated.

4.5. Experiments with the baseline system

In these experiments we employed the original HTK-based speech recogniser of the Saplen system only. Hence, the recognition results that were the input to the semantic analyser of the system were not corrected by the ASRPC

Table 4
Utterance corpora used for training and testing

Sentence type	Number of training utterances	Number of test utterances
Product order	250	250
Telephone number	250	250
Postal code	250	250
Address	250	250
Query	125	125
Confirmation	125	125
Number	125	125
Food name	125	125
Ingredient	125	125
Drink name	125	125
Size	125	125
Taste	125	125
Temperature	125	125
Street name	125	125
Building number	125	125
Building floor	125	125
Apartment letter	125	125
Error indication	125	125
Total	2750	2750

Table 5
Progress of learnt information as the size of training corpus increases

Number of dialogues	Number of patterns in α	Number of lexical confusions in β
150	146	408
300	161	551
450	167	684
600	172	763
750	175	908
900	175	1042
1050	175	1042

Table 6
Average evaluation results (%) of the baseline system

Dialogue corpus	WA	SU	IR	TC
DialoguesA$_1$	76.12	54.71	9.19	24.51

module. Employing *ScenariosA* and the user simulator we generated a corpus of 900 dialogues that we called *DialoguesA*$_1$. To create these dialogues the simulator used each scenario 3 times, behaving as a very cooperative user, a cooperative user, and a not very cooperative user. Table 6 sets out the average results obtained from the analysis of this dialogue corpus.

The low WA rate shows the problems of the speech recogniser in correctly analysing the utterances. These problems affected the other evaluation measures, which were very low. Observing the created log files (see Fig. 6) we found that in some cases the recognition results were incorrect but similar to the input utterances. For example, “una cerveza grande” (one large beer) was recognised as: “una cerveza grandes”, which was correctly understood by the system because of its implicit recovery ability. We found that 9.19% of the incorrectly recognised sentences were correctly understood by the system due to this reason. In other cases, the recognition results were completely different to the utterances and it was impossible for the system to implicitly recover the errors. For example, the address: “calle Acera de Canasteros numero uno, tercero, letra e” (Acerca de Canasteros Street, number one, third foor, letter e) was recognised as: “e de cereza hacer un pedido necesito de bazan” (e of cherry order I need of bazan).

4.6. Experiments with the proposed technique

4.6.1. Creation of the elements for applying the technique

In accordance with Section 3.1.1, we created a set of word-classes $\Omega = \{K_1, K_2, \ldots, K_{21}\}$ by re-using the 21 word-classes available from a previous study (López-Cózar and Callejas, 2005). These classes were the same word-classes used by the semantic analyser of the Saplen system. According to Section 3.1.2, we created a set of grammatical rules that contained three rules to check number correspondences in food and drink orders uttered in Spanish. To create the sets of syntactic–semantic and lexical models, as discussed in Sections 3.1.3 and 3.1.4, we used *DialoguesA*$_1$ and obtained $\alpha = \{\mathrm{SSM}_{\mathrm{T}i}\}$ and $\beta = \{\mathrm{LM}_{\mathrm{T}i}\}$, $i = 1, \ldots, 43$ (43 is the number of different prompt types generated by the Saplen system).

4.6.2. Decision on the optimal value for the similarity threshold

To decide the best value for the similarity threshold t discussed in Section 3.2.1, we carried out side experiments testing six different values: 0.3, 0.4, 0.5, 0.6, 0.7 and 0.8. We used *ScenariosB* (100 dialogues) and employing the user simulator we generated six dialogue corpora, one per value of t. The user simulator employed each scenario 3 times, simulating very cooperative, cooperative and not very cooperative users. Hence, each corpus contained 300 dialogues. The Saplen system used the ASRPC module during this corpora generation, employing the α and β sets created as discussed in the previous section. Analysis of the 6 corpora showed that the best performance of the system was achieved when $t = 0.5$.

4.6.3. Using the proposed technique with the optimal value of the similarity threshold

In these experiments the output of the HTK-based speech recogniser was processed by the ASRPC module (see Fig. 6). The similarity threshold was set to the optimal value: $t = 0.5$. We used again *ScenariosA* (300 scenarios) and employing the user simulator we generated 900 dialogues, which we called *DialoguesA*$_2$. As in the previous experiments, each scenario was used three times, one per user type. Table 7 shows the average results obtained from the analysis of this dialogue corpus.

Analysis of the log files showed that the ASRPC module was successful in correcting some erroneous recognition results. Table 8 shows some examples of these corrections, where T = prompt type of the Saplen system, U = user utterance, h = speech recognition result, and h' = result of the ASRPC module. The corrections were carried out by replacing or discarding words in h.

The technique worked very well for correcting errors in affirmative or negative confirmations. This happened because the speech recogniser usually substituted the word ‘si’ (yes) employed in many affirmative confirmations by the word ‘seis’ (six), especially when ‘si’ was uttered by speakers with strong southern Spanish accents. These users generally omitted the final ‘s’ of words, thus making ‘seis’ to be acoustically very similar to ‘si’. Also because of these accents, the recogniser usually substituted the word ‘no’ by the word ‘dos’. The ASRPC module corrected both kinds of error by replacing the concept NUMBER with the concept CONFIRMATION, and then selecting the most likely word in the latter concept given the word ‘seis’ (in affirmative confirmations) or ‘dos’ (in negative confirmations).

The technique also corrected many incorrectly recognised product orders. For example, “dos fantas grandes de limon” (two large lemon fantas) was recognised as “uno fantas grandes de limon” (one large lemon fantas) because of the acoustic similarity between ‘dos’ en ‘uno’ when uttered by strongly accented speakers. The ASRPC module repaired the error doing no corrections at the statistical level, and replacing ‘uno’ with ‘dos’ at the linguistic level. In other cases the correction was carried out at the statistical level. For example, “una ensalada de curry” (one curry salad) was recognised as “una error ensalada de curry”

Table 7
Average evaluation results (%) employing the proposed technique

Dialogue corpus	WA	SU	IR	TC
DialoguesA$_1$	84.62	71.25	13.20	68.32

Table 8
Examples of successful corrections of speech recognition errors (in Spanish)

T = PRODUCT_ORDER
U = **dos** fantas grandes de limon
h = **uno** (0.5954) fantas (1.0000) grandes (0.8987) de (0.9011) limon (1.0000)
h' = **dos** (1.0000) fantas (1.0000) grandes (0.8987) de (0.9011) limon (1.0000)

T = POSTAL_CODE
U = dieciocho cero cero **uno**
h = dieciocho (1.0000) cero (1.0000) cero (1.000) **pavo** (0.3000)
h' = dieciocho (1.0000) cero (1.0000) cero (1.000) **uno** (1.0000)

T = TELEPHONE_NUMBER
U = plaza alonso cano **numero** dieciocho cuarto letra a
h = plaza (0.8000) alonso (1.0000) cano (1.0000) **normal** (0.4000) dieciocho (1.0000) cuarto (0.5000) letra (0.6358) a (0.64320)
h' = plaza (0.8000) alonso (1.0000) cano (1.0000) **numero** (0.4000) dieciocho (1.0000) cuarto (0.5000) letra (0.6358) a (0.64320)

T = ANYTHING_TO_DRINK
U = **no**
h = **dos** (0.4233)
h' = **no** (1.0000)

T = TELEPHONE_CONFIRMATION
U = nueve cinco **ocho** sesenta setenta ochentinueve
h = nueve (0.3999) cinco (1.0000) **kas** (1.0000) sesenta (1.0000) setenta (1.0000) ochentinueve (1.0000)
h' = nueve (0.3999) cinco (1.0000) **ocho** (1.0000) sesenta (1.0000) setenta (1.0000) ochentinueve (1.0000)

T = FOOD_ORDER_CONFIRMATION
U = **una** ensalada de curry
h = **una** (0.8982) **error** (0.6950) ensalada (0.5982) de (0.5969) curry (0.8059)
h' = **una** (0.8982) ensalada (0.5982) de (0.5969) curry (0.8059)

T = TELEPHONE_CONFIRMATION
U = **si**
h = **seis** (0.8623)
h' = **si** (1.0000)

T = TELEPHONE_CONFIRMATION
U = **nueve** cinco ocho veintiuno catorce dieciocho
h = **dame** (0.8562) cinco (0.9632) ocho (.0856) veintiuno (0.1000) catorce (0.1000) dieciocho (0.9854)
h' =**nueve** (1.0000) cinco (0.9632) ocho (.0856) veintiuno (0.1000) catorce (0.1000) dieciocho (0.9854)

T = ADDRESS
U = calle almona del boquerón numero cinco segundo **letra h**
h = calle (1.0000) almona (1.0000) del (1.0000) boqueron (1.0000) **error** (0.5003) cinco (0.9000) segundo (0.6002) **cero** (0.7995)
h' = calle (1.0000) almona (1.0000) del (1.0000) boqueron (1.0000) **numero** (1.0000) cinco (0.9000) segundo (0.6002) **h** (1.0000)

T = ADDRESS
U = calle arandas **numero** ocho **primero** letra c
h = calle (0.4014) arandas (1.0000) **normal** (1.0000) ocho (1.0000) **primera** (0.6998) letra (0.7145) c (0.8510)
h' = calle (0.4014) arandas (1.0000) **numero** (1.0000) ocho (1.0000) **primera** (0.6998) letra (0.7145) c (0.8510)

(one mistake curry salad). The error correction was carried out by discarding the word in the concept ERROR (‘error’).

The technique was also able to correct some misrecognised telephone numbers. For example, “nueve cinco ocho veintiuno catorce dieciocho” (nine five eight twenty-one fourteen eighteen) was recognised as “dame cinco ocho veintiuno catorce dieciocho” (gimme five eight twenty-one fourteen eighteen) because of the acoustic similarity between ‘nueve’ and ‘dame’. The ASRPC module corrected the error by replacing the concept DESIRE with the concept NUMBER, and selecting the most likely word in the latter concept (‘nueve’) given the word ‘dame’.

The technique was also useful in repairing some misrecognised postal codes. For example, “dieciocho cero cero uno” (eighteen zero zero one) was recognised as “dieciocho cero cero pavo” (eighteen zero zero turkey). This error was corrected by replacing the concept INGREDIENT with the concept NUMBER, and selecting the most likely word in the latter concept (‘uno’) given the word ‘pavo’.

The ASRPC module was also successful in correcting some incorrectly recognised addresses. For example, “calle almona del boquerón numero cinco segundo letra h” (almona del boqueron street number five second[8] letter h) was recognised as “calle almona del boqueron error cinco segundo cero” (almona del boqueron street error five second zero). The error was corrected by making a double repair. First, replacing the concept ERROR with the

[8] In this sentence in Spanish, the word ‘floor’ was implicit after the word ‘second’, i.e. the data for the system was ‘second floor’.

concept NUMBER_ID, and selecting the most likely word in the latter concept ('numero') given the word 'error'. Second, replacing the concept NUMBER with the concept LETTER, and selecting the most likely word in the latter concept ('h') given the word 'zero'.

4.6.4. Advantage of using contextual syntactic–semantic models (SSM_T's)

The goal of this experiment was to check whether using different $\mathrm{SSM_T}$'s, and if needed $\alpha = \{\mathrm{SSM}_{Ti}\}$ as a fallback strategy, as discussed in Section 3.2.1, was preferable to the two following alternative strategies:

(i) use α only, without first checking the prompt-dependent $\mathrm{SSM_T}$'s;
(ii) use $\mathrm{SSM_T}$, and if the pattern is not found there use α as a fallback strategy, but without considering[9] the similarity threshold t.

The α set was created with *DialoguesA*$_1$ in Section 4.6.3, and the similarity threshold was set to the optimal value, $t = 0.5$. We first changed slightly the behaviour of the ASRPC module so that it worked in accordance with the procedure described in (i). We used again *ScenariosA* (300 scenarios) and the user simulator to generate a corpus of 900 dialogues, which we called *DialoguesA*$_3$ (each scenario was used three times, one per user type). Next we changed again the behaviour of the ASRPC module so that it now worked in accordance with the procedure described in (ii). We used again *ScenariosA* and the user simulator and generated another corpus of 900 dialogues, which we called *DialoguesA*$_4$ (each scenario was used three times, one per user type). Therefore, *DialoguesA*$_1$, *DialoguesA*$_3$ and *DialoguesA*$_4$ were created using the same scenarios and were comprised of the same number of dialogues, the only difference being in the strategy for selecting the syntactic–semantic model to be used. Table 9 shows the average results obtained from the analysis of *DialoguesA*$_3$ and *DialoguesA*$_4$.

Analysis of the log files showed that the error correction in confirmations was very much affected by the strategy employed to select the correction model (either $\mathrm{SSM_T}$ or α). This selection had a considerable effect on dialogue success because correctly recognising confirmations is critical for the dialogue. If we always used $\mathrm{SSM_T}$ to correct errors in confirmations, the correction was in many cases successful. On the other hand, if we always used α the correction was mostly incorrect. This happened because the pattern comprised of just the concept NUMBER (i.e. *ssp* = NUMBER) was in α. Therefore, if the answer to a confirmation prompt was 'si' (yes) and it was incorrectly recognised as 'seis' (six), the input pattern obtained from the recognition

[9] Note that the proposed strategy takes into account the similarity threshold t. The fallback strategy, i.e. use of the α set, is employed only if no patterns are found in $\mathrm{SSM_T}$ similar to the input pattern with *similarity* value greater than t.

Table 9
Average evaluation results (%) obtained by changing the strategy to select the syntactic–semantic model

Dialogue corpus	WA	SU	IR	TC
DialoguesA$_3$ (using α only)	80.15	61.67	9.57	39.78
DialoguesA$_4$ (using $\mathrm{SSM_T}$ if possible, otherwise use α)	82.26	66.84	12.15	55.35

result was: *ssp* = NUMBER, which matched one pattern in α. Hence, the algorithm for error correction did not make any correction at the pattern matching step and the recognition result remained uncorrected. If we compare Tables 9 and 7 it follows that the best strategy for selecting the model for the conceptual correction (either $\mathrm{SSM_T}$ or α) is the one proposed in this paper, which makes the decision considering the similarity threshold.

4.6.5. Advantage of using contextual lexical models (LM_T's)

Analogously to the previous section, the goal of this experiment was to check whether using different $\mathrm{LM_T}$'s and if needed $\beta = \{\mathrm{LM}_{Ti}\}$ as a fallback strategy, as discussed in Section 3.2.2, was preferable to using β only. To carry out the experiment we used the β set created with *DialoguesA*$_1$ in Section 4.6.3. Similarly as we did in the previous section, we changed slightly the behaviour of the ASRPC module in accordance with the new strategy, i.e. use β always instead of different $\mathrm{LM_T}$'s. We employed again *ScenariosA* (300 scenarios) and the user simulator and generated a corpus of 900 dialogues, which we called *DialoguesA*$_5$ (each scenario was used three times, one per user type). Therefore, *DialoguesA*$_1$ and *DialoguesA*$_5$ were obtained using the same scenarios and were comprised of the same number of dialogues, the only difference being in the use of β. Table 10 shows the average results obtained from the analysis of *DialoguesA*$_5$.

These results are lower than those shown in Table 7, which indicates that using the lexical information (word confusions) associated with each prompt type T ($\mathrm{LM_T}$) for correction is better than using all the lexical information in the application domain regardless of the prompt type. This happens because the confusion probabilities of words are not the same in the $\mathrm{LM_T}$'s and in β, and these differences are in some cases deterministic in making the proper correction. For example, in accordance with the β set used in the experiments, the highest probability of confusing the word 'error' with a word in the NUMBER concept is 0.0370, and this word is 'dieciseis' (16). However, in accordance with $\mathrm{LM_{T=PRODUCT\_ORDER}}$, the highest probability of confusing the word 'error' with a word in the

Table 10
Average evaluation results (%) obtained by changing the strategy to select the lexical model

Dialogue corpus	WA	SU	IR	TC
DialoguesA$_5$ (using β only)	81.40	65.61	11.43	60.89

NUMBER concept is 0.0090, and this word is 'una' (one). Therefore, if in the correction of a recognition result we assume that the word 'error' is incorrect, and we look for a word in the NUMBER concept, the selected candidate is 'dieciseis' if we consider β, and 'dos' if we take into account $LM_{T=PRODUCT\_ORDER}$. This shows that the lexical model employed affects the outcome of the correction process. If we compare Tables 10 and 7 it follows that the best strategy to select the model for the lexical correction (either LM_T or β) is the one proposed in this paper, which uses different models to take into account information about word confusions in different contexts.

5. Limitations of the proposed technique

The technique has several limitations. One is that the system designers must provide semantic information about the application domain in the form of word-classes, and linguistic information in the form of grammatical rules. This is not the case for purely statistical methods, which learn from the provided training data without requiring additional effort on the part of the system designers. Another limitation is that there are cases where the technique fails in correcting erroneous recognition results, as observed in Table 11. One failing case is when the words in the input utterance are substituted by others and the recognition result is valid in the application domain. For example, the product order "quiero una ensalada de gambas" (I want one prawn salad) was recognised as "quiero una ensalada de manzana" (I want an apple salad). In this case the ASRPC module did not made any correction because the recognition result was conceptually valid although it was incorrect. Another example is the telephone number "nueve cinco ocho setenticuatro setenticinco veintiuno" (nine five eight seventy-four seventy-five twenty-one), which was recognised as "nueve cinco ocho setenticuatro sesenticinco veintiuno" (nine five eight seventy-four sixty-five twenty-one) because of the substitution of 'setenticinco' by 'sesenticinco'. Again, the ASRPC module did

Table 11
Examples of unsuccessful corrections of speech recognition errors (in Spanish)

T = TELEPHONE_NUMBER
U = nueve cinco ocho setenticuatro **setenticinco veintiuno**
h = nueve (0.5999) cinco (1.0000) ocho (1.0000) setenticuatro (0.8000) **sesenticinco** (0.4000) **veintidos** (1.0000)
h' = nueve (0.5999) cinco (1.0000) ocho (1.0000) setenticuatro (0.8000) **sesenticinco** (0.4000) **veintidos** (1.0000)

T = CONFIRM_TELEPHONE_NUMBER
U = nueve cinco ocho setenticuatro **setenticinco** veintiuno
h = **dame** (0.6006) cinco (1.0000) ocho (1.0000) setenticuatro (0.5000) **veinticinco** (0.1999) veintiuno (1.0000)
h' = **nueve** (1.0000) cinco (1.0000) ocho (1.0000) setenticuatro (0.5000) **veinticinco** (0.1999) veintiuno (1.0000)

T = ADDRESS
U = **calle acera del triunfo número dos** segundo letra a
h = **c** (0.7983) **diez** (0.2010) **noveno** (0.2013) segundo (0.6038) letra (0.6188) a (0.6038)
h' = **c** (0.7983) **diez** (0.2010) **noveno** (0.2013) segundo (0.6038) letra (0.6188) a (0.6038)

T = FOOD_ORDER
U = **ponme una fanta de naranja grande**
h = **queso** (0.1010) de (0.9005) **bazan** (0.7013) **tercera** (0.7013)
h' = **queso** (0.1010) de (0.9005) **bazan** (0.7013) **tercera** (0.7013)

T = TELEPHONE_NUMBER
U = nueve cinco ocho **tres dos** cero tres uno siete
h = nueve (0.6450) cinco (0.7941) ocho (0.6019) **queso** (0.4002) cero (1.000) **tres** (0.8998) uno (1.0000) **siete** (1.0000)
h' = nueve (0.6450) cinco (0.7941) ocho (0.6019) **cero** (0.5392) tres (0.8998) uno (1.0000)

T = TELEPHONE_NUMBER
U = nueve cinco **ocho uno tres cinco uno cinco seis**
h = nueve (0.7002) cinco (0.9000) **chocolate** (0.5995) **curry** (0.6994) **trece** (0.5995)
h' = nueve (0.7002) cinco (0.9000) **trece** (0.5995)

T = ARDES
U = **calle alhóndiga número cuatro**
h = **queremos** (0.5013) **de** (0.8056) **lomo** (0.4063) **con** (0.4063) **queso** (0.4063)
h' = **queremos** (0.5013) **de** (0.8056) **lomo** (0.4063) **con** (0.4063) **queso** (0.4063)

T = PRODUCT_ORDER
U = quiero una ensalada de **gambas**
h = quiero (0.5056) una (1.0000) ensalada (0.9012) de (0.9005) **manzana** (0.6924)
h' = quiero (0.5056) una (1.0000) ensalada (0.9012) de (0.9005) **manzana** (0.6924)

T = PRODUCT_ORDER
U = **quiero una ensalada de gambas**
h = de (1.0000) **manzana** (1.0000) **manzana** (0.5008)
h' = de (1.0000) **manzana** (1.0000) **manzana** (0.5008)

not make any correction because the obtained telephone number was conceptually valid although it was misrecognised.

Another failing case is when the sequence of words in the input utterance is so distorted by speech recognition errors that the obtained syntactic–semantic pattern cannot be corrected. In this case the technique either fails in the correction or does not make any correction. For example, the product order "ponme una fanta de naranja grande" (I want one large orange fanta) was recognised as "queso de bazan tercera" (cheese of bazan third). For this very incorrectly recognised sentence the ASRPC did not make any correction, as it did not find any similar pattern learnt in the training to make concept replacements.

Another limitation is the well-known problem of out-of-vocabulary words (OOV), which exists whenever a speech recogniser is used. All the words uttered by the speakers who recorded the speech database were included in the dictionary of the Saplen system employed in the experiments. Hence, there were no OOV problems in the testing of the technique. However, in a real interaction there may be OOV words. The effect of these words in the proposed technique depends on whether these words are keywords or not, and on the speech recognition errors that they cause.

If the OOV words are not keywords it is not important for the dialogue system to recognise the words, as they do not affect the semantics of the sentences. However, the uttering of these words may cause in-vocabulary words to be inserted, substituted or deleted in the recognition results. In terms of speech understanding, there is no problem if these in-vocabulary words are not keywords as the concepts in the utterances are not changed. The problem is when the words are keywords. The performance of the technique in dealing with in-vocabulary keyword recognition errors has been discussed in Section 3.2.3.

If the OOV words are keywords it would be important for the dialogue system to recognise the words, given that they affect the semantics of the sentences. However, these words cannot be recognised as they are unknown for the recogniser, and thus the speech recognition errors they cause cannot be corrected. This happens because the syntactic–semantic and lexical models obtained from the training cannot learn any information about these words. Hence, it is not possible to make either concept or word replacements to correct the errors. It is impossible as well to make corrections at the linguistic level given that OOV words do not appear in the recognition results, and thus the grammatical rules are not applicable for them.

If we wanted to check the effect of OOV words on the proposed technique employing our experimental setting, we could remove some words from the Saplen system's dictionary, which would now be OOV words. Another method would be to record new utterances which include OOV words, and create new scenarios for the user simulator that include as goals the frames associated with these new utterances. The dialogue system would analyse the new utterances when the simulator employs the new scenarios. In both cases the speech recogniser would face the problem of unknown words, which would enable recognition errors to be handled by the technique.

6. Conclusions and future work

This paper has presented a novel technique for ASR post-correction in spoken dialogue systems that employs semantic, syntactic, lexical and contextual information. The semantic and syntactic information is implemented by means of sets of patterns, which are created from the analysis of a dialogue corpus. Each set of patterns is mapped to a prompt type of a spoken dialogue system. The prompt type represents the contextual information of the technique, as it determines the kinds of sentence likely to be uttered by the user at a particular dialogue state. The lexical information is implemented by means of word confusion probabilities and basic grammatical rules. The confusion probabilities are computed from the analysis of the dialogue corpus, by aligning utterance transcriptions and speech recognition results. The grammatical rules are provided by the system designers and are used to ensure grammatical restrictions between specific words. These rules are necessary to compensate for problems arising from sparse training data.

The technique presents two main novel contributions to research. On the one hand, it considers several contexts where a speech recognition result can be corrected. A similarity threshold is used to decide whether the correction must be carried out in the context associated with the current prompt type generated by the dialogue system, or in another context. The optimal value of this threshold must be determined experimentally. On the other hand, the technique deals with the confidence scores of the words employed for the corrections. We have proposed a simple method to compute these scores that worked well in the experiments, but we do not claim that it is optimal.

Experiments to test the technique have been carried out employing a dialogue system and a user simulator, both developed in previous studies. The dialogue system used one prompt-independent language model (word bigram) for speech recognition, which enabled, in theory, the recognition of any kind of sentence permitted in the application domain, regardless of the current prompt of the dialogue system. We observed in previous experiments that this language model caused many erroneous recognition results. Hence, we used the proposed technique to try to correct some of these errors. We compared two speech recognition front-ends: (i) the HTK-based speech recogniser of the dialogue system (baseline), and (ii) the HTK-based recogniser and the ASRPC module, which implements the proposed technique. The results obtained, shown in Table 6 (baseline) and Table 7 (proposed technique), indicate that the technique was very useful in correcting speech recognition errors, as the system's word accuracy, speech understanding, implicit recovery and task completion rates increased by 8.5%, 16.54%, 4%, and 44.17%, respectively.

We also carried out experiments to check the effect on the correction process of using the contextual information provided by the system prompt in terms of syntactic–semantic models (SSM_T's) and lexical models (LM_T's). Regarding the former models, comparison of Tables 7 and 9 shows that using SSM_T's with a similarity threshold t provides better results than: (i) using the set of these models (α) only, and (ii) using the SSM_T's if the patterns are found there, otherwise using the α set as a fallback strategy without considering the similarity threshold. Regarding the lexical models, comparison of Tables 7 and 10 shows that using LM_T's if possible, and otherwise the set of these models (β) as a fallback strategy, is also preferable to using β only.

Future work includes testing the proposed technique with dialogue systems developed for other application domains, for example the Viajero system that we developed in a previous study to provide bus travel information (López-Cózar et al., 2000). Another line of research is concerned with studying other methods to decide the confidence score for the correction word. For example, one alternative strategy might be to consider the score proportional to the number of words in the concept that contains the word with which the correction word is confused. Another method would be to consider it proportional to the number of phonemes in common between the corrected word and the correction word.

Acknowledgements

The authors with to thank the reviewers for their valuable comments that have enhanced this paper.

References

Allen, J., 1995. Natural Language Understanding. The Benjamin/Cummings Publishing Company Inc..

Billi, R., Castagneri, G., Danielli, M., 1997. Field trial evaluations of two different information inquiry systems. Speech Comm. 23 (1–2), 83–93.

Crestani, F., 2000. Word recognition errors and relevance feedback in spoken query processing. In: Proc. Conf. on Flexible Query Answering Systems, pp. 267–281.

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

Denda, Y., Tanaka, T., Nakayama, M., Nishiura, T., Yamashita, Y., 2007. Noise-robust hands-free voice activity detection with adaptive zero crossing detection using talker direction estimation. In: Proc. ICSLP, pp. 222–225.

Fisher, W.M., Fiscus, J.G., 1993. Better alignment procedures for speech recognition evaluation. In: Proc. ICASSP, pp. 59–62.

Hazen, T.J., Seneff, S., Polifroni, J., 2002. Recognition confidence scoring and its use in speech understanding systems. Computer Speech Lang. 16, 49–67.

Huang, X., Acero, A., Hon, H., 2001. Spoken Language Processing: A Guide to Theory. In: Algorithm and System Development. Prentice-Hall, 2001.

Jeong, M., Kim., B., Lee, G.G., 1996. Semantic-oriented error correction for spoken query processing. In: Proc. ICSLP, pp. 897–900.

Jeong, M., Jung, S., Lee, G.G., 2004. Speech recognition error correction using maximum entropy language model. In: Proc. Interspeech, pp. 2137–2140.

Kaki, S., Sumita, E., Iida, H., 1998. A method for correcting speech recognitions using the statistical features of character co-occurrences. In: Proc. COLING-ACL, pp. 653–657.

Kakutani, N., Kitaoka, N., Nakagawa, S., 2002. Detection and recognition of repaired speech on misrecognized utterances for speech input of car navigation system. In: Proc. ICSLP, pp. 833–836.

Kellner, A., Rueber, B., Seide, F., Tran, B.-H., 1997. PADIS – an automatic telephone switchboard and directory information system. Speech Comm. 23, 95–111.

Kraiss, K.F. (Ed.), 2006. Advanced Man–Machine Interaction: Fundamentals and Implementation. Springer.

Lee, C.-H., Carpenter, B., Chou, W., Chu-Carroll, J., Reichl, W., Saad, A., Zhou, Q., 2000. On natural language call routing. Speech Comm. 31, 309–320.

Levow, G.A., 1998. Characterizing and recognizing spoken corrections in human–computer dialogue. In: Proc. COLING-ACL, pp. 736–742.

Levow, G.-A., 2002. Adaptation in spoken corrections: implications for models of conversational speech. Speech Comm. 36, 147–163.

Lo, W.K., Soong, F.K., 2005. Generalized posterior probability for minimum error verification of recognized sentences. In: Proc. ICASSP, pp. 85–88.

López-Cózar, R., Araki, M., 2005. Spoken, Multilingual and Multimodal Dialogue Systems. In: Development and Assessment. John Wiley & Sons Publishers.

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

López-Cózar, R., García, P., Díaz, J., Rubio, A.J., 1997. A voice activated dialogue system for fast-food restaurant applications. In: Proc. Eurospeech, pp. 1783–1786.

López-Cózar, R., Rubio, A.J., García, P., Segura, J.C., 1998. A spoken dialogue system based on a dialogue corpus analysis. In: Proc. LREC, pp. 55–58.

López-Cózar, R., Rubio, A.J., García, P., Díaz-Verdejo, J.E., López-Soler, J.M., 2000. Sistema de diálogo oral para proporcionar información sobre viajes en autobús (Spoken dialogue system to provide bus information). In: Proc. First Spanish Workshop on Speech Technologies, Seville, Spain.

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

López-Cózar, R., Callejas, Z., McTear, M.F., 2006. Testing the performance of spoken dialogue systems by means of a new artificially-simulated user. Artif. Intell. Rev. 26, 291–323.

Mangu, L., Padmanabhan, M., 2001. Error corrective mechanisms for speech recognition. In: Proc. ICASSP, pp. 29–32.

McTear, M.F., 2004. Spoken Dialogue Technology. Toward the Conversational User Interface. Springer.

Morales, N., Gu, L., Gao, Y., 2007. Adding noise to improve noise robustness in speech recognition. In: Proc. ICSLP, pp. 930–933.

Nakano, N., Minami, Y., Seneff, S., Hazen, T.J., Cyphers, D.S., Glass, J., Polifroni, J., Zue, V., 2001. Mokusei: a telephone-based Japanese conversational system in the weather domain. In: Proc. Eurospeech, pp. 1331–1334.

Ogata, J., Goto, M., 2005. Speech repair: quick error correction just by using selection operation for speech input interfaces. In: Proc. Interspeech, pp. 133–136.

Rabiner, L., Juang, B.H., 1993. Fundamentals of Speech Recognition. Prentice-Hall.

Ringger, E.K., Allen, J.F., 1996. A fertility model for post correction of continuous speech recognition. In: Proc. ICSLP, pp. 897–900.

Seneff, S., Polifroni, J., 2000. Dialogue management in the Mercury flight reservation system. In: Proc. ANLP-NAACL Satellite Workshop, pp. 1–6.

Seto, S., Kanazawa, H., Shinchi, H., Takebayashi, Y., 1994. Spontaneous speech dialogue system TOSBURG II and its evaluation. Speech Comm. 15, 341–353.

Shi, Y., Zhou, L., 2006. Examining knowledge sources for human error correction. In: Proc. Interspeech, pp. 1089–1092.

Skantze, G., 2005. Exploring human error recovery strategies: implications for spoken dialogue systems. Speech Comm. 45, 325–341.

Suhm, B., Myers, B., Waibel, A., 2001. Multimodal error correction for speech user interfaces. ACM Trans. Computer–Human Inter. 8 (1), 60–98.

Swerts, M., Litman, D., Hirschberg, J., 2000. Correction in spoken dialogue system. In: Proc. ICSLP, pp. 615–618.

Wahlster, W. (Ed.), 2006. SmartKom: Foundations of Multimodal Dialogue Systems. Springer.

Ward, W., Issar, S., 1996. A class based model for speech recognition. In: Proc. ICASSP, pp. 416–418.

Zhou, Z., Meng, H., 2004. A two-level schemata for detecting recognition errors. In: Proc. ICSLP 2004, pp. 449–452.

Zhou, Z., Meng, H., Lo, W.K., 2006. A multi-pass error detection and correction framework for Mandarin LVCSR. In: Proc. ICSLP, pp. 1646–1649.

Zue, V., Seneff, S., Glass, J., Polifroni, J., Pao, C., Hazen, T., Hetherington, L., 2000. Jupiter: a telephone-based conversational interface for weather information. IEEE Trans. Speech Audio Process. 8 (1), 85–96.