New Technique to Enhance the Performance of Spoken Dialogue Systems by Means of Implicit Recovery of ASR Errors

Ramón López-Cózar[1], David Griol[2], and José F. Quesada[3]

[1] Dept. of Languages and Computer Systems, CITIC-UGR, University of Granada, Spain
rlopezc@ugr.es

[2] Dept. of Computer Science, Carlos III University of Madrid, Spain
dgriol@inf.uc3m.es

[3] Dept. of Artificial Intelligence and Computer Science, University of Seville, Spain
jose.quesada@infinity.es

Abstract. This paper proposes a new technique to implicitly correct some ASR errors made by spoken dialogue systems, which is implemented at two levels: statistical and linguistic. The goal of the former level is to employ for the correction knowledge extracted from the analysis of a training corpus comprised of utterances and their corresponding ASR results. The outcome of the analysis is a set of syntactic-semantic models and a set of lexical models, which are optimally selected during the correction. The goal of the correction at the linguistic level is to repair errors not detected during the statistical level which affects the semantics of the sentences. Experiments carried out with a previously-developed spoken dialogue system for the fast food domain indicate that the technique allows enhancing word accuracy, spoken language understanding and task completion by 8.5%, 16.54% and 44.17% absolute, respectively.

1 Introduction

It is well known that user utterances are frequently misheard, misrecognised or misunderstood by spoken dialogue systems (SDSs), mainly due to current limitations of state-of-the-art automatic speech recognition (ASR). Thus, well designed error handling strategies are crucial in providing robust system performance, specially for not experienced users. Some authors have studied human error recovery strategies in order to apply these, if possible, to SDSs. For example, [1] found that when subjects face speech recognition problems, a common strategy is to ask task-related questions that confirm their hypothesis instead of signalling non-understanding, which leads to better understanding of subsequent sentences.

A number of error handling strategies can be found in the literature, mostly working a three levels of the system's architecture: ASR, spoken language understanding

* This research has been funded by the Spanish Ministry of Science and Technology, under project TIN2007-64718 HADA.

(SLU) and dialogue management (DM). These techniques have been traditionally separated into two groups: error *detection* and *correction*.

A common method for error detection is using recognition confidence scores, but the problem is that these measures are not entirely reliable, as depend on noise conditions and user types. At the DM level, a common method for detecting errors is using confirmation strategies [2] or re-phrasing, whereas implicit confirmations can be employed for error detection and correction.

[3] proposed a model for error correction, which is comprised of four levels: detection, diagnosis, repair plan selection and plan execution interactively. [4] presented an agent-based architecture in which error handling is divided into individual, application independent components. This architecture makes it possible to construct adaptive and reusable components and entire error-handling toolkits. [5] proposed an example-based error recovery method to detect and correct errors, based on a re-phrase strategy and a task guidance to help novice users re-phrase well-recognisable and well-understandable sentences.

Some error handling techniques try to hide errors made by some components of SDSs, for example, errors made by the ASR. The technique that we propose in this paper follows this direction, as its goal is to detect ASR errors and correct them before the ASR result is the input to the SLU. To do so, it takes the ASR result and carries out two kinds of process, one statistical and the other linguistic. The first process tries to detect and correct errors by considering knowledge extracted from the analysis of a training corpus comprised of utterances and their corresponding ASR results. The goal of the linguistic process is to repair errors not detected during the statistical process, if these affect the semantics of the sentences.

The remainder of the paper is organised as follows. In section 2 we discuss previous studies concerned with error handling techniques to automatically detect and correct ASR errors. Section 3 presents our technique for implicit recovery of these errors. It discusses the necessary elements and explains how to implement the algorithms for error correction. Section 4 presents the experiments, comparing system performance results achieved with and without using the proposed technique. Finally, section 5 presents the conclusions and outlines possibilities for future work.

2 Related Work

Most previous techniques for automatically correcting ASR errors are based on statistical methods that use probabilistic information about words uttered and words in the recognition results. For example, following this approach, [6] proposed a method based on two parts. On the one hand, a channel model represents errors made by a speech recogniser, whilst on the other, a language model represents the likelihood of a sequence of words uttered by one 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.

Following this approach, [7] proposed a method that uses statistical features of characters co-occurrence, which was implemented in two consecutive correcting processes. The former detects and corrects errors using a database of erroneous-correct

utterance pairs. The remaining errors are passed to the second process, which uses a string in the corpus that is similar to the string including recognition errors. The authors found that this method made significant contributions to the performance of speech translation systems.

Also, [8] proposed a method to model likely contexts of all words in an ASR system vocabulary by performing a lexical co-occurrence analysis using a large corpus of output from a speech recogniser. They identified regions in the data that contain likely contexts for a given query word. Finally, they detected words or sequences of words that are likely to appear in the context and that are phonetically similar to the query word. Their experiments proved that this method provides high-precision in detection and correction of errors.

Several authors have proposed carrying out error detection and correction at several levels. For example, [9] proposed a two-level schema for detecting recognition errors. The first level applies an utterance classifier to decide whether the speech recognition result is erroneous. If it is decided to be incorrect, it is passed to the second level where a word classifier is used to decide which words are misrecognitions.

Following the same approach, [10] proposed a method for error detection based on three levels. The first is to detect whether the input utterance is correct. The second level is to detect incorrect words, and the third level is to detect erroneous characters. The error correction first creates candidate lists of errors, and then re-ranks the candidates with a combined model of mutual information and trigram.

Statistical methods 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 erroneous strings, since they are directly dependent on the lexical entries. Hence, a number of authors have proposed to combine different types of information sources. For example, [11] presented a method that models the output generated by a set of ASR systems as independent knowledge sources that can be combined and used to generate an output with reduced error rate. The outputs of the single ASR systems are combined into a minimal cost word transition network by means of iterative applications of dynamic programming alignments. The resulting network is decided employing a rescoring or *voting* process that selects the output sequence with the lowest score.

Employing a different approach, [12] combined lexical information with higher level knowledge sources via a maximum entropy language model (MELM). Error correction was arranged at 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 main problem found was that the training of the MELM required a lot of time and was sometimes infeasible.

3 Proposed Technique for Implicit Recovery from ASR Errors

In this paper we propose a new technique to enhance the performance of SDSs using a method that implicitly recovers from some ASR errors. We state that the recovery is *implicit* as the user is not aware of the error, in other words, no dialogue turns are employed for error recovery which makes a more natural and friendly interaction with the system. This technique is inspired by previous studies based on pattern matching [7] and statistical information [9], sorting out one drawback of the former type of methods, namely, that the selected pattern may not be optimal. To address this limitation, our technique employs several corpora of previously learnt syntactic-semantic and lexical patterns, as well as a similarity threshold $d \in [0.0 - 1.0]$ to decide whether one pattern is good enough for error correction. If it is found not to be good enough, the technique searches for a better pattern in the whole set of patterns available, and if a proper pattern is not found there either, the technique does not make any correction using the patterns. This procedure will be discussed in detail in Section 3.5.

In the following sections we describe the elements required to implement our technique: concepts, grammatical rules, syntactic-semantic models and lexical models.

3.1 Concepts

We define a *concept* as a set of keywords of a given type which are necessary to extract the semantic content of sentences within an application domain. For example, in our experiments in the fast food domain that will be described in section 4, we consider, among others, the following concepts: DESIRE = {want, need, …}, FOOD = {sandwich, cake, salad, …}, DRINK = {water, beer, wine, …} and AMOUNT = {one, two, three, …}.

3.2 Grammatical Rules

The general format of a grammatical rule is as follows: *ssp* → *restriction*, where *ssp* denotes a syntactic-semantic pattern, which will be described in the following section, and *restriction* is a condition that must be satisfied by all the concepts in the pattern. For example, one rule used in our experiments in Spanish is:

NUMBER DRINK SIZE →
number(NUMBER) = *number*(DRINK) **and**
number(DRINK) = *number*(SIZE) **and**
number(NUMBER) = *number*(SIZE)

where *number* is a function that returns either *singular* or *plural* for each Spanish word in the concepts that it uses as input. The goal of this rule is to check number correspondences of drink orders uttered in Spanish. For example, the sentence *dos cervezas grandes* (two large beers) holds this correspondence.

3.3 Syntactic-Semantic Models

A syntactic-semantic model is a conceptual representation of the sentences uttered by users of a SDS in a dialogue state. This state is associated with a prompt type T of the dialogue system, which represents equivalent prompts to obtain a particular data type from the user. To create a syntactic-semantic model for a prompt type T, we transform each sentence uttered by the user in response to the prompt type into what we call a *syntactic-semantic pattern* (*ssp*). This pattern is a sequence of concepts obtained by replacing each word in the sentence with the concept(s) the word belongs to. From the analysis of all the sentences uttered in response to each prompt type we create a set of *ssp*'s, in which we remove those that are redundant and associate with each *ssp* its relative frequency within the set. The outcome of this process is a syntactic-semantic model associated with the prompt type T (SSM_T). We call α model the set of SSM_T's created considering the m prompt types of a SDS: $\alpha = \{SSM_{Ti}\}$, $i = 1 \ldots m$.

3.4 Lexical Models

Lexical models contain information about the performance of the speech recogniser of a SDS. We must create a lexical model for each prompt type T, which we call LM_T. To do so, we consider the sentences uttered in response to the prompt type and their corresponding recognition results. The format of this model is: $LM_T = \{w_a, w_b, p_{ab}\}$, where w_a is a word uttered by a user, w_b is the recognised word and p_{ab} is the posterior probability of obtaining w_b given w_a. To create LM_T we align each uttered sentence with the recognised sentence using the method described in [13], and compute the probabilities p_{ab} for each word pair (w_a, w_b). We call β model the set of LM_T's created considering the m prompt types of a SDS: $\beta = \{LM_{Ti}\}$, $i = 1 \ldots m$.

3.5 Algorithms to Implement the Technique

In this section we discuss the two levels for error detection and correction employed by our technique: statistical and linguistic.

3.5.1 Correction at the Statistical Level

The goal of this correction level is to find words w_I's in the recognised sentence which belong to incorrect concepts K_I's. For each word, we must decide the correct concept K_C and select the most appropriate word $w_C \in K_C$ to substitute w_I in the recognised sentence. We can implement this procedure in two steps: pattern matching and pattern alignment.

3.5.1.1 Pattern Matching. The procedure for pattern matching is illustrated in Fig. 1. It receives as input the recognised sentence: w_1, w_2, w_3, w_4, … w_n, the syntactic-semantic model associated with the current prompt type T, and the similarity threshold d.

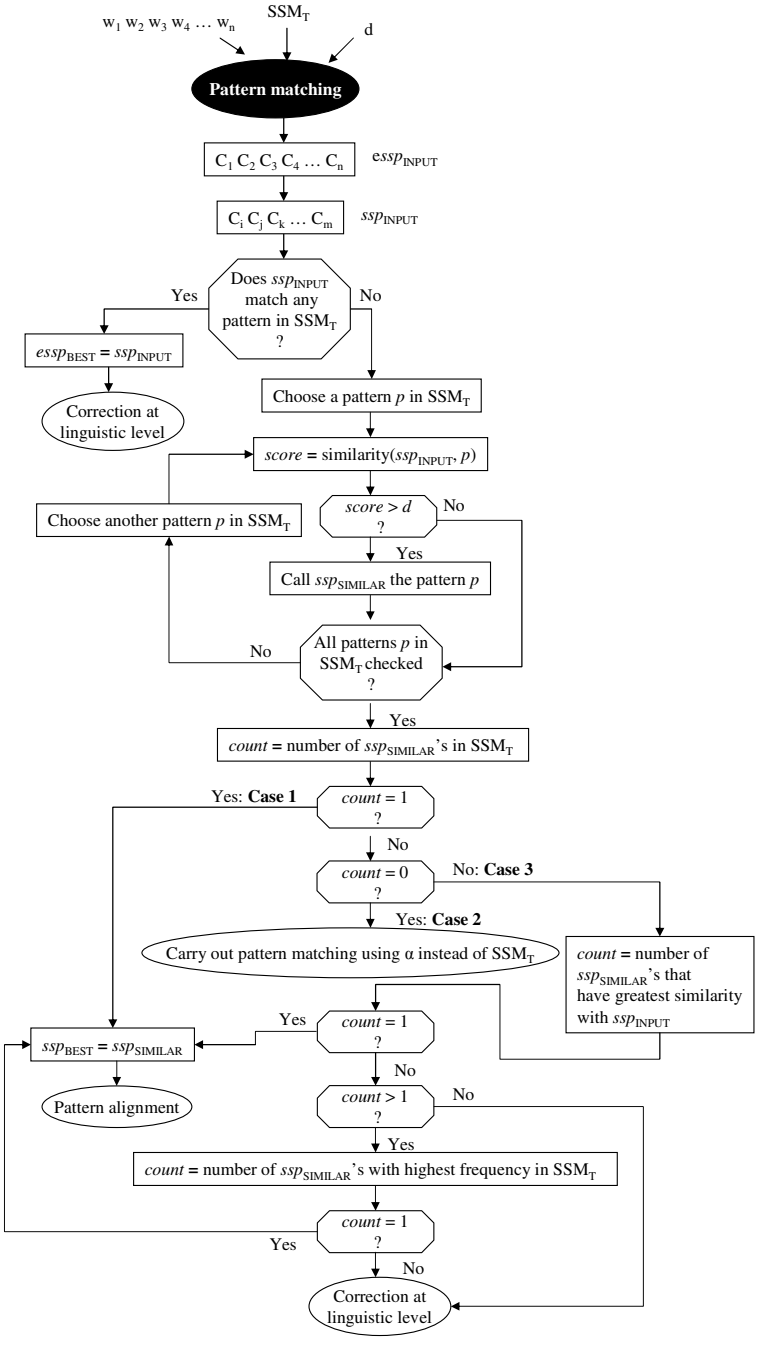

Fig. 1. Pattern matching for correction at the statistical level

The procedure employs what we call an *enriched syntactic-semantic pattern* (*essp*$_{\text{INPUT}}$) obtained from the recognised sentence. This pattern is a sequence of what we call *containers*: C_1, C_2, C_3, … C_n. The goal of this step is to transform *essp*$_{\text{INPUT}}$ into another pattern concept sequence called *essp*$_{\text{BEST}}$, which is initially empty. To do this, we firstly create a syntactic-semantic pattern called *ssp*$_{\text{INPUT}}$, which only contains the concepts in *essp*$_{\text{INPUT}}$, for example: *ssp*$_{\text{INPUT}}$ = DESIRE AMOUNT INGREDIENT FOOD. Secondly, we determine whether *ssp*$_{\text{INPUT}}$ matches any pattern in the syntactic-semantic model associated with the prompt type T (SSM$_T$). If so, we make *essp*$_{\text{BEST}}$ = *essp*$_{\text{INPUT}}$ and proceed with the correction at the linguistic level (which will be discussed in section 3.5.2). Otherwise, we look for patterns *p* similar to *ssp*$_{\text{INPUT}}$ in SSM$_T$. To do this, we compare *ssp*$_{\text{INPUT}}$ with every pattern *p* in the model, and compute a similarity score as follows: $similarity(ssp_{\text{INPUT}}, p) = (n - m_{ed}) / n$, where *n* is the number of concepts in *ssp*$_{\text{INPUT}}$ and m_{ed} is the minimum edit distance between both patterns, computed using the method described in [14]. We call *ssp*$_{\text{SIMILAR}}$ any pattern *p* in SSM$_T$ such that $similarity(ssp_{\text{INPUT}}, p) > d$, where $d \in [0.0, 1.0]$ is a similarity threshold, the optimal value of which must be experimentally determined.

We consider three cases depending on the number of *ssp*$_{\text{SIMILAR}}$'s in SSM$_T$. The first case is when there is just one. In this case, we create a new pattern called *ssp*$_{\text{BEST}}$, make *ssp*$_{\text{BEST}}$ = *ssp*$_{\text{SIMILAR}}$ and proceed with the pattern alignment, which will be discussed in section 3.5.1.2.

The second case is when there are no *ssp*$_{\text{SIMILAR}}$'s in SSM$_T$. In this case, we try to find *ssp*$_{\text{SIMILAR}}$'s in the α model (discussed in section 3.3) instead of doing so in SSM$_T$, i.e., we employ the same procedure but considering α, not SSM$_T$.

The third case is when there are several *ssp*$_{\text{SIMILAR}}$'s in SSM$_T$ (or in α). The question then is to determine the best *ssp*$_{\text{SIMILAR}}$. To make this selection we search for the *ssp*$_{\text{SIMILAR}}$ that has the greatest similarity with *ssp*$_{\text{INPUT}}$. If there is just one *ssp*$_{\text{SIMILAR}}$ satisfying this condition, we make *ssp*$_{\text{BEST}}$ = *ssp*$_{\text{SIMILAR}}$ and proceed with Step 2 (pattern alignment). If there are several patterns, we select those with the highest frequency in SSM$_T$ (or in α): if there is just one, we make *ssp*$_{\text{BEST}}$ = *ssp*$_{\text{SIMILAR}}$ and proceed with Step 2; if there are several we do not make any correction at the statistical level.

3.5.1.2 Pattern Alignment. The goal of pattern alignment is to build *essp*$_{\text{BEST}}$ in case it is still empty. The procedure is illustrated in Fig. 2. It receives as input the pattern concept sequence *essp*$_{\text{BEST}}$, the lexical model associated with the current prompt type T (LM$_T$) and the syntactic-semantic patterns *ssp*$_{\text{INPUT}}$ and *ssp*$_{\text{BEST}}$.

The procedure takes into account each container C_a in *ssp*$_{\text{INPUT}}$ and considers two cases. The first is when the word w_a in C_a does not affect the semantics of the sentence, i.e., it is not a keyword (e.g. *please*). In this case we create a new container *D*, make $D = C_a$ and add *D* to *essp*$_{\text{BEST}}$.

The second case is when the word w_a in C_a affects the semantics of the sentence, i.e., it is a keyword (e.g. *sandwich*). Thus, we study whether the word must be corrected. To do this, we try to align the container C_a with a container C_b in *ssp*$_{\text{BEST}}$ using the method described in [13] and consider two possible occurrences:

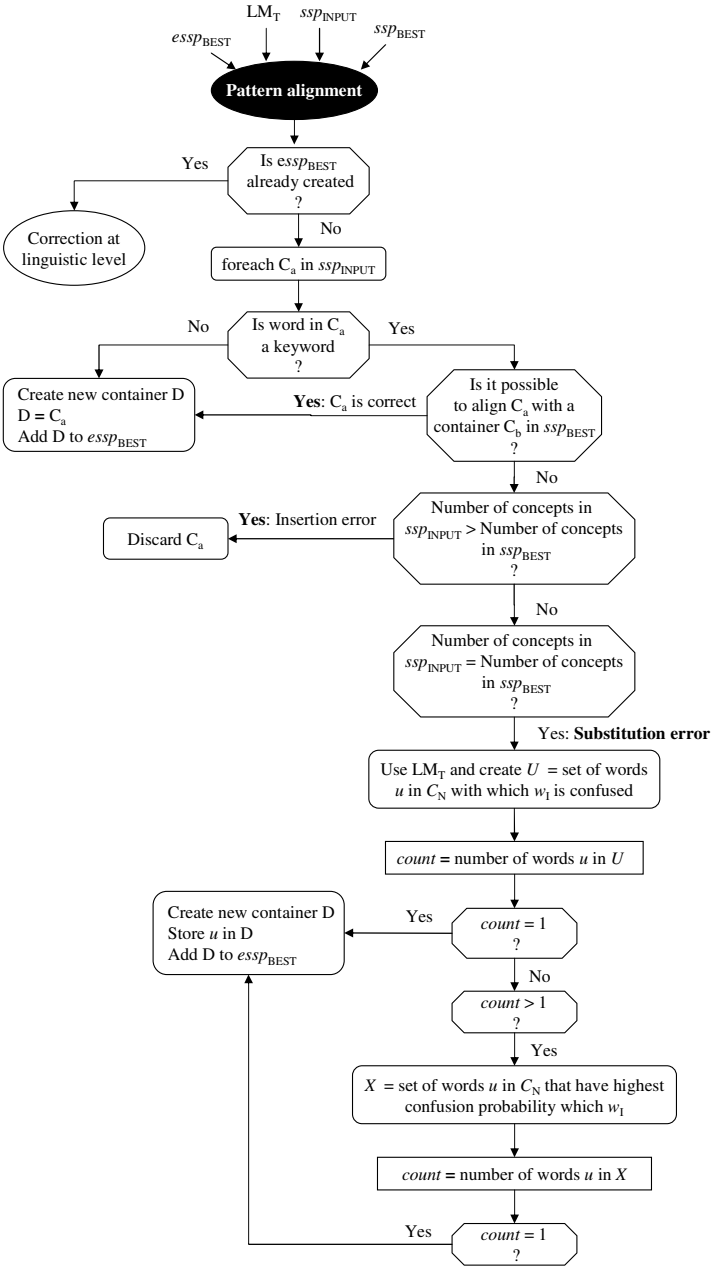

Fig. 2. Pattern alignment for correction at the statistical level

Occurrence 1: C_a can be aligned. In this occurrence we assume that the container C_a is correct and do not make any correction at the statistical level. We create a new container D, make $D = C_a$ and add D to $essp_{BEST}$.

Occurrence 2: It is not possible to align C_a. This occurrence may happen in two situations. The first is when the container is a result of an insertion recognition error. In this situation we discard C_a, i.e. it is not added to $essp_{BEST}$. The second situation is when the container is a result of a substitution recognition error. Therefore, we must find a correction word from a different concept, $w_C \in C_N$, store it in a new container D, and add this container to $essp_{BEST}$. To find w_C we consider the lexical model associated with the prompt type T (LM_T) and create the set U of words $u \in C_N$ with which the word w_I is confused. If there is only one word u in U, we create a new container D that we name C_N, store it in u, and add D to $essp_{BEST}$. If there are several words, we carry out the same procedure but using the word that has the highest confusion probability with w_I if it is unique; if it is not unique, or there are no words in U, we do not make any correction at the statistical level.

3.5.2 Correction at the Linguistic Level

The goal of the correction at the linguistic level is to repair errors that are not detected at the statistical level and affect the semantics of sentences. To carry out the correction we use the grammatical rules described in section 3.2. For each rule we carry out the following procedure. The syntactic-semantic pattern *ssp* of the rule is inserted in a *window* that slides from left to right over $essp_{BEST}$, as can be observed in Fig. 3.

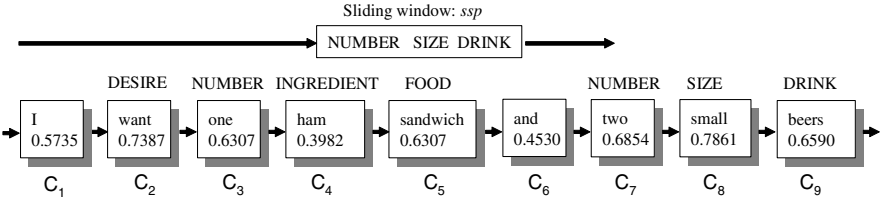

Fig. 3. Sliding window over $essp_{BEST}$

If the concept sequence in the window is found in $essp_{BEST}$, then we apply the *restriction* of the rule to the words in the containers of $essp_{BEST}$. If the words satisfy the restriction, we do not make any correction. Otherwise, we try to find out the reason for the insatisfaction by searching for an incorrect word w_I. To decide the word w_C to correct the incorrect word, we consider the lexical model LM_T and take into account the set $U = \{u_1, u_2, ..., u_p\}$ comprised of words of the same concept than the word w_I. Next, we proceed similarly as discussed in the second case of Occurrence 2 (see previous section) but considering that the goal now is to replace one word in one concept with other word in the same concept.

4 Experiments

The goal of the experiments described in this section is to test the proposed technique using the Saplen spoken dialogue system, which we developed in a previous study to answer fast food queries and orders made in Spanish [15]. The evaluation has been carried out in terms of word accuracy (WA), spoken language understanding (SLU) and task completion (TC), considering two front-ends for ASR: i) *baseline ASR*, comprised of the standard HTK-based speech recogniser of the Saplen system, and ii) *enhanced ASR*, comprised of the same speech recogniser plus an additional module that implements the proposed technique.

We have employed a dialogue corpus collected in our University from students interacting with the Saplen system, which contains around 5,500 utterances and roughly 2,000 different words. The utterance corpus has been divided into two separate corpora, each containing around 50% of the utterances. Using the training corpus we have compiled a word bigram that allows recognising sentences of the 18 different types in the corpus. The remaining 50% of the utterances have been used for testing.

The experiments have been carried out employing a user simulator developed in a previous study [16]. The interaction between the Saplen system and the simulator is decided considering a set of scenarios that represent user goals. We have created two scenario sets: *ScenariosA* (300 scenarios) and *ScenariosB* (100 scenarios). Each dialogue generated by the interaction between the Saplen system and the user simulator is stored in a log file for analysis and evaluation purposes.

Given that the construction of the syntactic-semantic and lexical models described in sections 3.3 and 3.4 has been carried out employing simulated dialogues, we have made additional experiments to decide the necessary number of dialogues to obtain the maximum amount of syntactic-semantic and lexical knowledge. The results indicate that 900 dialogues is the optimal trade-off.

4.1 Experiments with the Baseline ASR

Employing the user simulator, the Saplen system and *ScenariosA,* we have generated a corpus of 900 dialogues, which we have called *Dialogues*A_1. Table 1 sets out the average results obtained from the analysis of this corpus. The results show the problems of the system in correctly recognising and understanding some utterances. Analysis of the log files reveals that in some cases the misrecognised sentences are similar to the uttered sentences. For example, the sentence in Spanish *dos fantas grandes de limón* (two large lemon fantas) is recognised as *uno fantas grandes de limón* (one large lemon fantas) because of the acoustic similarly between *dos* and *uno* when uttered by users with strong Southern Spanish accents.

Table 1. Results using the baseline ASR (in %)

WA	SLU	TC
76.12	54.71	24.51

We have also observed problems with confirmations, which happen because the speech recogniser usually substitutes the word *si* (yes) by the word *seis* (six), when the former word is uttered by strongly accented speakers. In other cases, the recognised sentences are very distorted by ASR errors. For example, the sentence *quiero una fanta de naranja grande* (I want one big orange Fanta) is sometimes recognised as *queso de manzana tercera* (cheese of apple third).

4.2 Experiments with the Enhanced ASR

As the *concepts* required for the technique (discussed in section 3.1), we have employed a set of 21 *concepts* that we created in a previous study [15]. Following section 3.2 we have created a set of grammatical rules to check the number correspondences for food and drink orders. To create the syntactic-semantic and lexical models, discussed in sections 3.3 and 3.4, we have analysed $DialoguesA_1$ thus obtaining $\alpha = \{SSM_{Ti}\}$ and $\beta = \{LM_{Ti}\}$, with i = 1 ... 43, given that the Saplen system can generate 43 different prompt types.

To decide the optimal value for the similarity threshold d (discussed in section 3.5.1) we have carried out experiments considering values in the range [0.1, 0.9]. Employing the user simulator and *ScenariosB*, we have generated a corpus comprised of 300 dialogues for each value, using in all cases the proposed technique. Analysis of the outcomes of these experiments reveals that the best results are obtained when d = 0.5. Using this optimal value, we have employed again *ScenariosA* to generate another corpus of 900 dialogues, which we call $DialoguesA_2$. Table 2 shows the average results obtained from the analysis of this corpus.

Table 2. Results using the enhanced ASR (in %)

WA	SLU	TC
84.62	71.25	68.32

Analysis of the log files shows that the technique is successful in correcting some incorrectly recognised sentences. For example, the incorrectly recognised drink order *one large lemon fantas* is corrected by doing no changes at the syntactic-semantic level, and replacing *one* with *two* at the lexical level. In other product orders the correction is carried out at the semantic-syntactic level. For example, *one curry salad* is sometimes recognised as *one error curry salad*. In this case the correction is carried out removing the ERROR concept at the syntactic-semantic level.

The technique is useful in correcting the errors with confirmations discussed in the previous section. To do this, it replaces the NUMBER concept with the CONFIRMATION concept, and then selects the most likely word in CONFIRMATION.

The enhanced ASR enables as well correction of some misrecognised telephone numbers (in the Spanish format). For example, *nine five eight twenty-one fourteen eighteen* is sometimes recognised as *gimme five eight twenty-one fourteen eighteen* because of acoustic similarity between *nine* and *gimme* in Spanish. The technique corrects the error by replacing the DESIRE concept with the NUMBER concept and selecting the most likely word in NUMBER given the word *gimme* at the lexical level.

The technique is also useful to correct some misrecognised postal codes. For example, *eighteen zero zero one* is sometimes recognised as *eighteen zero zero turkey*. This error is corrected by replacing the INGREDIENT concept with the NUMBER concept and selecting the most likely word in NUMBER given the word *turkey*.

Our proposal is also successful in correcting some incorrectly recognised addresses (in the Spanish format). For example, *almona del boquerón street number five second floor letter h* is sometimes recognised as *almona del boquerón street error five second floor letter zero*. This error is corrected by making a double correction. First, replacement of the ERROR concept with the NUMBER_ID concept and selection of the most likely word in NUMBER_ID given the word *error*. Second, replacement of the NUMBER concept with the LETTER concept and selection of the most likely word in LETTER given the word *zero*.

There are cases where the technique fails in detecting errors, and thus in correcting them. This happens when words in the uttered sentence are substituted by other words and the result is valid in the application domain. For example, this occurs when the sentence *two green salads* is recognised as *twelve green salads*, given that there is no conflict in terms of *concepts* and there is agreement in number between the words.

4.2.1 Advantage of Using SSM_T's, α and *d*

In this experiment we have checked whether using SSM_T's or α, taking into account *d*, is preferable to the two following alternative strategies: i) use α only without firstly checking the SSM_T's, and ii) use the SSM_T's, but if the pattern ssp_{INPUT} is not found in these, use α without considering the similarity threshold *d*. The α model is the one created employing $DialoguesA_1$ and *d* is set to the optimal value, i.e., $d = 0.5$. We have implemented strategy i) and used *ScenariosA* to generate a corpus of 900 dialogues, which we call $DialoguesA_3$. Next, we have implement strategy ii) and, using again *ScenariosA*, have generated another corpus of 900 dialogues, which we call $DialoguesA_4$. Therefore, $DialoguesA_1$, $DialoguesA_3$ and $DialoguesA_4$ have been created using the same scenarios and are comprised of the same number of dialogues, the only difference being in the strategy for selecting the correction model to be used. Table 3 shows the average results obtained from the analysis of $DialoguesA_3$ and $DialoguesA_4$.

Table 3. Results employing strategies to select the syntactic-semantic correction model (in %)

Corpus	WA	SLU	TC
$DialoguesA_3$	80.15	61.67	39.78
$DialoguesA_4$	82.26	66.84	55.35

Analysis of the log files shows that the error correction in confirmations is very much affected by the strategy employed to select the correction model (either SSM_T or α). If we always use SSM_T to correct errors in confirmations, the correction is in many cases successful. On the other hand, if we always use α the correction is mostly incorrect.

4.2.2 Advantage of Using LM_T's, β and *d*

The goal of this experiment has been to check whether using the LM_T's or β taking into account *d* is preferable to using β regardless of *d*. To carry out the experiment we have used the β model created with *DialoguesA*$_1$. We have employed again *ScenariosA* and generated a corpus of 900 dialogues, which we call *DialoguesA*$_5$. Therefore, *DialoguesA*$_1$ and *DialoguesA*$_5$ have been obtained using the same scenarios and are comprised of the same number of dialogues, the only difference being in the use of β. Table 4 shows the average results obtained from the analysis of *DialoguesA*$_5$. The experiment shows that the confusion probabilities of words are not the same in the LM_T's and β. For example, considering the β model, the highest probability of confusing the word *error* with a word in the NUMBER concept is 0.0370, and this word is *dieciseis* (sixteen). However, considering $LM_{T=PRODUCT\text{-}ORDER}$, this probability is 0.0090 and the word is *una* (one). Therefore, the correction word is *dieciseis* if we consider β, and *una* if we take into account $LM_{T=PRODUCT\text{-}ORDER}$, which in some cases is deterministic in making the proper correction.

Table 4. Results employing an alternative strategy to select the lexical model (in %)

Corpus	WA	SLU	TC
DialoguesA$_5$	81.40	65.61	60.89

5 Conclusions and Future Work

Comparing the results set out in Tables 1 and 2 we observe that the proposed technique allows enhancing the performance of the Saplen system in terms of WA, SLU and TC by 8.5%, 16.54% and 44.17% absolute, respectively. These enhancements are mostly achieved because considering the proposed threshold for similarity scores between patterns, the technique decides whether to use correction models associated with the current prompt type T (SSM_T and LM_T), or general correction models for the application domain (α and β). This novel contribution optimises the procedure for error recovery, as can be observed from comparison of results set out in Tables 2, 3 and 4. These results show that our method for selecting the correction models is preferable to other possible strategies for selecting these models. In particular, we have observed that the benefit of the proposed method is particularly noticeable in the correction of misrecognised confirmations.

Future work includes considering additional information sources to correct errors that in the current implementation cannot be detected, such as domain-dependent knowledge. For example, in our application domain we could use this kind of information to consider that the sentence *twelve green salads*, although syntactically correct, is likely to be incorrectly recognised, given that it is not usual that customers of fast food restaurants order such a large amount of a product. We also plan to study the performance of the technique considering prompt-dependent similarity thresholds.

References

1. Skantze, G.: Exploring human error recovery strategies: Implications for spoken dialogue systems. Speech Communication 45, 325–341 (2005)
2. McTear, M., O'Neill, I., Hanna, P., Liu, X.: Handling errors and determining confirmation strategies – An object-based approach. Speech Communication 45, 249–269 (2005)
3. Duff, D., Gates, B., Luperfoy, S.: An architecture for spoken dialogue management. In: Proc. of ICSLP, pp. 1025–1028 (1996)
4. Turunen, M., Hakulinen, J.: Agent-based error handling in spoken dialogue systems. In: Proc. of Eurospeech, pp. 2189–2192 (2001)
5. Lee, C., Jung, S., Lee, D., Lee, G.G.: Example-based error recovery strategy for spoken dialog system. In: Proc. of ASRU, pp. 538–543 (2007)
6. Ringger, E.K., Allen, J.F.: A fertility model for post correction of continuous speech recognition. In: Proc. of ICSLP, pp. 897–900 (1996)
7. Kaki, S., Sumita, E., Iida, H.: A method for correcting speech recognitions using the statistical features of character co-occurrences. In: Proc. of COLING-ACL, pp. 653–657 (1998)
8. Sarma, A., Palmer, D.D.: Context-based speech recognition error detection and correction. In: Proc. of HLT-NAAACL, pp. 85–88 (2004)
9. Zhou, Z., Meng, H.: A two-level schemata for detecting recognition errors. In: Proc. of ICSLP, pp. 449–452 (2004)
10. Zhou, Z., Meng, H., Lo, W.K.: A multi-pass error detection and correction framework for Mandarin LVCSR. In: Proc. of ICSLP, pp. 1646–1649 (2006)
11. Fiscus, J.G.: A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (Rover), pp. 347–352 (1997)
12. Jeong, M., Jung, S., Lee, G.G.: Speech recognition error correction using maximum entropy language model. In: Proc. of Interspeech, pp. 2137–2140 (2004)
13. Fisher, W.M., Fiscus, J.G.: Better alignment procedures for speech recognition evaluation. In: Proc. ICASSP, pp. 59–62 (1993)
14. Crestani, F.: Word recognition errors and relevance feedback in spoken query processing. In: Proc. Conf. on Flexible Query Answering Systems, pp. 267–281 (2000)
15. López-Cózar, R., Callejas, Z.: Combining language models in the input interface of a spoken dialogue system. Computer Speech and Language 20, 420–440 (2006)
16. López-Cózar, R., de la Torre, A., Segura, J.C., Rubio, A.J., Sánchez, V.: Assessment of dialogue systems by means of a new simulation technique. Speech Communication 40(3), 387–407 (2003)