Enhancing Speech Understanding in Spoken Dialogue Systems by Means of a New Frame-Correction Technique

Ramón López-Cózar¹, Zoraida Callejas¹, David Griol²

¹Dept. of Languages and Computer Systems, University of Granada, Spain
²Dept. of Computer Science, Carlos III University of Madrid, Spain

{rlopezc, zoraida}@ugr.es ; dgriol@inf.uc3m.es

Abstract

This paper proposes a new technique to enhance speech understanding in spoken dialogue systems, which aims to replace semantic frames incorrectly generated by the systems with the correct ones. To do so, it relies on a training procedure that takes into account previous system misunderstandings for each dialogue state. Experiments have been carried out employing two systems (Saplen and Viajero) previously developed in our lab, which employ a prompt-independent language model and several prompt-dependent language models for ASR. The results show that the technique enhances system performance for both kinds of language model, especially for the prompt-independent language model. Using this technique, Saplen increases sentence understanding by 19.54%, task completion by 26.25% and word accuracy by 7.53%, whereas for Viajero these figures increase by 14.93%, 18.06% and 6.98%, respectively.

Index Terms: Spoken dialogue systems, speech recognition, speech understanding, dialogue management.

1. Introduction and Related Work

Humans have an extraordinary ability to understand speech even in adverse conditions due to noise, bandwidth limitations, hearing disabilities, or psychological/physical problems. This ability relies on the use of several knowledge sources that most of the times are unconsciously handled. For example, we can replace perceived sounds that do not correspond to any known word to try to recover from errors in the perception process. Lexical, syntactic, semantic and pragmatic knowledge is used together with paralinguistic information (e.g., gestures, facial expressions and body movements of the speaker) to try to infer the conveyed spoken message.

Giving the current limitations of state-of-the-art automatic speech recognition (ASR), spoken dialogue systems (SDSs) sometimes fail in recognising and understanding what the user says. This problem has caused that many potential users find this technology unreliable. To address this problem, it is necessary to develop techniques for ASR, spoken language understanding (SLU) and dialogue management that increase system robustness in real-world conditions. For example, a number of previous studies have focused on ASR, considering that the better the recognition, the better the understanding [1, 2]. Other authors have focused on dialogue management, allowing flexibility in user interaction in an attempt to handle unexpected contributions and to interpret them correctly within the dialogue context [3].

In this paper we focus on enhancing speech understanding and present a technique that considers the dialog context with the aim of modelling the human ability to deal with misunderstandings. Our technique has similarities with [4], which employs a pattern matching phase and two rewriting phases. The first rewriting phase has as input the recognised sentence and produces as output a sequence of semantic constraints. Basically, it obtains a semantic representation from the syntactic patterns in the input.

A difference between the two techniques is that ours takes as its input a frame generated by the system’s SLU component and produces another frame as the output. The second rewriting phase of [4] applies heuristic rewriting rules of four types: object merging, constraint inference, filtering and abstraction. The filtering rule can be considered related to our algorithm to remove inadequate tuples from the correction model, which will be explained in section 2.2.2, whereas the constraint inference rule can be considered related to our algorithm to generalise the behaviour of the frame correction module, which will be discussed in section 2.2.3.

2. The Proposed Technique

Fig. 1 shows the architecture of the robust speech understanding module that implements the proposed technique. This module is comprised of the SLU module of a SDS and a new module that carries out frame corrections. The latter uses the current system prompt and what we call a correction model (CM) to decide how to correct misunderstandings caused by ASR errors. The correction is carried out by replacing the incorrect frame generated by the system’s SLU module with another frame that is assumed to be correct considering the system prompt.

This research has been funded by Spanish project ASIES TIN2010-17344.
An advantage of this technique is that it is independent of the task performed by the system, and thus, it can be easily applied to systems designed for any application domain. For example, in the experiments we have applied the technique to two systems previously developed in our lab, one for fast food ordering (Saplen) [5] and the other for bus travel booking (Viajero) [6]. To set up the technique for a given dialogue system, the system designers must create and enhance a correction model, as explained in the following sections.

2.1. Creation of initial correction model

The initial correction model stores incorrect frames generated by the SLU module of the system as it processes the input utterances. It is comprised of tuples of the form: \((T, f_R, f_O)\), where \(T\) denotes a prompt type of the system, \(f_R\) represents the reference frame associated with the sentence uttered to answer the prompt, and \(f_O\) denotes the frame obtained by the SLU module of the system as it analyses the recognised sentence.

We consider that \(f_O\) is correct if it matches exactly \(f_R\), and is incorrect otherwise.

To create the correction model for our experiments, we used a procedure that takes as input each prompt type (\(T\)) of the dialogue system, the reference frame (\(f_R\)) associated with the utterance selected by a user simulator developed in a previous study [7], and the frame obtained (\(f_O\)) from the analysis of the recognised sentence. If \(f_R\) did not match \(f_O\), the procedure included the tuple \((T, f_R, f_O)\) into the correction model. It must be noted that the simulator selected utterances from a corpus collected from telephone calls of real users interacting with the system. Hence, the simulation allowed checking the performance of the system’s speech recogniser and SLU component as if the system was interacting with real users.

2.2. Enhancement of the initial correction model

The second task to implement the proposed technique is to optimise the initial correction model to obtain the model that the frame correction module will use (see Fig. 1). This task can be performed by carrying out three steps: compaction, removal of inadequate tuples, and generalisation of behaviour.

2.2.1. Compaction

The first step is to compact the initial correction model as it may contain too many repeated tuples. Hence, the goal is to reduce its size as much as possible to avoid any processing delay caused by the frame correction module. This step can be easily automated by means of a simple procedure that takes each tuple \((T, f_R, f_O)\) and looks for the same tuple in the model, removing duplicates if found.

2.2.2. Removal of inadequate tuples

The second step is to analyse the tuples in the already compacted correction model to prevent the frame correction module from replacing frames incorrectly. This can be done automatically following the algorithm shown in Fig. 2, which takes as input the already compacted correction model (CM) as well as the models that we call \(\Sigma\) and \(\Pi\), producing as output an enhanced version of the correction model. The \(\Sigma\) and \(\Pi\) models must be created in advance by the system designers, applying their knowledge about the application domain and system performance.

The \(\Sigma\) model stores information about the frames used by a dialogue system in a given application domain. It specifies which frame slots must be filled-in and which ones can be left empty in order to consider an obtained frame complete. For example, in our experiments a complete food order has filled-in the slots: amount, ingredients and food name. Hence, the order “one ham sandwich” would be considered complete whereas the order “sandwich” would not.

The \(\Pi\) model contains information about pairs of the form: \((\text{promptType}, \text{typeOfObtainedFrame})\), which represent expected prompt-answer pairs in the application domain. For example, using the Saplen or Viajero systems users typically utter their address when the system prompts them to do so. Thus, the pair \((\text{Address}?, \text{Address})\) must be included into the \(\Pi\) model.

![Fig. 2. Algorithm to remove inadequate tuples from the correction model.](image_url)

To remove inadequate tuples, the algorithm firstly analyses each tuple \((T, f_R, f_O)\) in the correction model employing a function called \(\text{Complete}\), which takes \(T\) and \(f_O\) and using the \(\Sigma\) model decides whether \(f_O\) is complete. If it is not complete, the tuple is removed from the model because it might be obtained from a variety of different inputs, and thus allowing it into the model may lead to incorrect frame replacements.

Secondly, the algorithm uses a function called \(\text{Agreement}\) to decide whether the type of \(f_O\) matches the prompt type \(T\) taking into account the \(\Pi\) model. If they match, the tuple is removed from the model. This is needed because an obtained frame may be incorrect regarding the reference frame but be correct for other input utterances. For example, an understood user address may be incorrect regarding the reference frame, but still be a valid, correct address. Hence, the corresponding tuple must be removed from the correction model to avoid potential incorrect frame replacements.
The final part of the algorithm checks that there are no tuples with the same prompt type $T$ and obtained frame $f_O$, which differ in the reference frame $f_R$, as this would mean that the obtained frame could be corrected in several ways. Hence, if these tuples are found, they are removed from the correction model to prevent incorrect frame replacements.

### 2.2.3. Generalisation of behaviour

The third step is to expand the correction model, now free from inadequate tuples, to generalise the behaviour of the frame correction module so that it can deal appropriately with misunderstandings not observed in the training. For example, if the model does not have any correction for the prompt: “Do you want to cancel the conversation and start again?”; the goal of this step is to include corrections for the possible misunderstanding of responses to this prompt, thus enabling the frame correction module to correct the errors. The algorithm for implementing this step takes as input the already available correction model and a model that we call $\Psi$, which contains classes of prompt types.

To create the $\Psi$ model the system designers must take into account all the possible prompt types of the system, and group them into classes considering as a classification criterion that all the prompt types in a class must have the same expected kind of response from the user. The algorithm firstly copies the tuples in the correction model CM to a new model CM$'$ that is initially empty. Next, it takes into account each tuple $(T, f_R, f_O)$ in CM and determines the class of prompt type (K) that contains T. The algorithm then checks whether tuples of the form $(T', f_R, f_O)$ with $T' \neq T$ are in CM, where $T'$ represents each prompt type in K. If a tuple $(T', f_R, f_O)$ is not in CM$'$ then it is added to this model. Finally, a new correction model CM is created containing the tuples in CM$'$. This model is the input to the frame correction module as shown in Fig. 1.

### 3. Experiments

The goal of the experiments was to test whether the proposed technique was useful to enhance sentence understanding (SU) and task completion (TC) of the Saplen and Viajero systems, which we previously developed in our lab for interacting with users in Spanish. Additionally, we wanted to check the effect, if any, on word accuracy (WA). Both systems employed an HTK-based speech recogniser and word bigrams as the language model.

#### 3.1. Utterance corpora and scenarios

We created two separate utterance corpora (for training and testing) for each system ensuring that no training utterances were included in the test corpus. Both corpora contained the orthographic transcriptions of the utterances as well as the corresponding reference frames $(\hat{f}_R)$. The corpus for Saplen contained around 5,500 utterances; 50% of them were used for training and the remaining for testing. The corpus for Viajero contained around 5,800 utterances, which were also divided into training (50%) and test (50%).

To collect user interactions with the systems, we employed a user simulator that we developed in a previous study [7]. The simulator used the test utterances for interacting with the systems. We designed 250 scenarios for each system. The scenario goals were reference frames selected at random from the utterance corpus used for testing. In the case of Saplen, the frames corresponded to product orders, telephone numbers, post codes and addresses, whereas for Viajero they were concerned with travel bookings, telephone numbers and queries on travel schedules, price and duration.

#### 3.2. Language modelling for ASR

We employed the two kinds of language models (word bigrams) that we used in a previous study [8]. For Saplen these models were 17 prompt-dependent language models (PDLMs) and one prompt-independent language model (PILM), whereas for Viajero they were 16 PDLMs and one PILM. For each system, there was a PDLM associated with each different system's prompt type, which was used to recognise the utterance provided by the user simulator to answer the system prompt.

The goal of the PILM was to recognise any kind of sentence permitted in the application domain regardless of the current system prompt. In previous experiments we observed that the performance of the systems using this kind of language model deteriorates slightly, given the broader range of sentence types and the greater vocabulary to be considered. However, this language model was appropriate for users who tended to answer system prompts with any kind of sentence within the application domain.

It was interesting for us to test the proposed technique using both kinds of language model (PDLMs and PILM) because we plan to study ways to let the systems automatically select one or the other as an attempt to adapt their performance to the kind of user (more or less experienced) and the success of system-user interaction.

#### 3.3. Results

### 3.3.1. Experiments with the baseline systems

In these experiments, the robust speech understanding module shown in Fig. 1 was not used. Thus, the frames received by the dialogue manager of the Saplen and Viajero systems were not corrected by the frame correction module. To save time, we created an initial correction model (see section 2.1) for each system and language model in order to get information on system misunderstandings that later could be used to apply the proposed technique, as will be discussed in section 3.3.2.

Employing the user simulator to interact with each system, we generated 10 dialogues for each scenario and language model. Hence, a total of 10 x 250 x 2 = 5,000 dialogues were generated for each system. Table 1 sets out the average results obtained in terms of sentence understanding (SU), task completion (TC) and word accuracy (WA) from the analysis of these dialogues.

<table>
<thead>
<tr>
<th>System</th>
<th>Language models</th>
<th>SU</th>
<th>TC</th>
<th>WA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saplen</td>
<td>PDLMs</td>
<td>72.66</td>
<td>65.83</td>
<td>81.83</td>
</tr>
<tr>
<td></td>
<td>PILM</td>
<td>69.71</td>
<td>63.11</td>
<td>80.65</td>
</tr>
<tr>
<td>Viajero</td>
<td>PDLMs</td>
<td>83.91</td>
<td>78.04</td>
<td>87.95</td>
</tr>
<tr>
<td></td>
<td>PILM</td>
<td>80.25</td>
<td>75.21</td>
<td>84.19</td>
</tr>
</tbody>
</table>

It can be observed that in some cases the results were under 70%. This happens because the utterances were taken from a corpus collected from telephone calls of real users interacting with the systems. The quality of the recordings was poor in general since the calls were made in non-clean conditions. Moreover, the utterances were affected by typical phenomena.
of spontaneous speech, such as filled pauses, hesitations and repetitions of words. The table shows as well that the systems worked slightly better when the PDLMs were used. The reason is that using this language model the speech recogniser employed a word bigram compiled from training sentences of the appropriate type. In addition, the vocabulary considered using the PDLMs was much smaller than when the PILM was used.

3.3.2. Experiments with the proposed technique

In these experiments, the Saplen and Viajero systems used the robust speech understanding module shown in Fig. 1. Therefore, the frames generated by the SLU module of the systems were replaced with the correct ones if they were considered incorrect, which was done before they were used by the dialogue manager.

Following section 2.1, we used the correction model created in the worst case of the experiments described in section 3.3.1, which corresponded to the usage of the PILM. In the case of Saplen, this model contained 119,773 tuples, whereas in the case of Viajero it contained 101,602 tuples.

Following section 2.2, we compacted the initial correction models for the two systems to remove repeated tuples. Then, we removed the inadequate tuples from the compacted models. Finally, we expanded the models to generalise the behaviour of the frame correction module to prompt types not observed in the training. As a result, we obtained a correction model with 359 tuples for Saplen and another correction model for Viajero with 320 tuples.

In order to obtain experimental results using the robust speech understanding module with these models, we employed again the user simulator and generated another 10 dialogues for each scenario and language model, i.e. $10 \times 250 \times 2 = 5,000$ dialogues for each dialogue system. Table 2 shows the average results obtained.

Table 2. System performance employing the proposed technique (%).

<table>
<thead>
<tr>
<th>System</th>
<th>Language Models</th>
<th>SU</th>
<th>TC</th>
<th>WA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saplen</td>
<td>PDLMs</td>
<td>91.03</td>
<td>91.28</td>
<td>88.9</td>
</tr>
<tr>
<td></td>
<td>PILM</td>
<td>89.25</td>
<td>89.36</td>
<td>88.18</td>
</tr>
<tr>
<td>Viajero</td>
<td>PDLMs</td>
<td>98.21</td>
<td>95.90</td>
<td>94.55</td>
</tr>
<tr>
<td></td>
<td>PILM</td>
<td>95.18</td>
<td>93.27</td>
<td>91.17</td>
</tr>
</tbody>
</table>

It can be observed that the systems worked better than in the previous experiments. Analysis of the log files created by the frame correction module revealed that most corrections were made due to misunderstandings caused by the strong Southern Spanish accents of many speakers in the corpus. For example, we observed that when these speakers answered system’s “Yes/No” confirmation prompts with the utterance “Si” (yes) the recognition result in many cases was “Seis” (six), given that due to their accents these users typically omitted the final ‘s’ of words. The proposed technique allowed to correct this misunderstanding by replacing the incorrect frame obtained by the systems’ SLU module (amount = “6”) with the correct one (action = “accept”) before it was the input to the systems’ dialogue manager. Hence, the speech recognition error did not affect the subsequent dialogue.

4. Conclusions and Future Work

A comparison of Tables 1 and 2 shows that the proposed technique improves the performance of the baseline systems. Regarding Saplen, SU increases by 18.37% absolute for the PDLMs and by 19.54% absolute for the PILM. The improvement in terms of SU reflects a remarkable increment in terms of TC, which is 25.45% absolute for the PDLMs and 26.25% absolute for the PILM.

Interestingly, we can observe enhancements in WA even though the goal of the proposed technique was to enhance SU. Analysis of the log files created by the frame correction module reveals that this improvement is mostly consequence of misunderstandings of user answers to the confirmation prompts of the baseline systems, which force the systems to make more attempts to get data confirmed. The proposed technique sorts out this problem by properly replacing the incorrect frames, as discussed above. Consequently, when the proposed technique is used the systems need to make fewer attempts to get data confirmed, which reduces the number of misrecognised words and thus increases WA.

We think the results obtained in the experiments are promising but there are ways to improve the proposed technique. For example, we could study methods to extract more information from the initial correction model. In the current implementation, the technique removes the repeated tuples in the model to make it as small as possible to avoid any processing delay that could slow down the interactivity of the dialogue systems. However, knowing the number of duplicates of a tuple $(T, f, O)$ can be important as it can provide information about how often the dialogue system misunderstands a sentence uttered to answer a prompt type $T$. Therefore, it could be interesting to make a deeper analysis of the tuples in the initial correction model and use this information to try to enhance the frame replacement process.

5. References