Strategies to Overcome Problematic Input in a Spanish Dialogue System

Victoria Arranz, Núria Castell and Montserrat Civit

TALP Research Center, Universitat Politècnica de Catalunya
Campus Nord - C6, Jordi Girona, 1-3, 08034 Barcelona, Spain
{varranz,castell,civit}@talp.upc.es

Abstract. This paper focuses on the strategies adopted to tackle problematic input and ease communication between modules in a Spanish railway information dialogue system for spontaneous speech. The paper describes the design and tuning considerations followed by the understanding module, both from a language processing and semantic information extraction point of view. Such strategies aim to handle the problematic input received from the speech recogniser, which is due to spontaneous speech as well as recognition errors.

1 Introduction

The nineties witnessed a boosting in spoken dialogue system development. The need for machines offering more natural and efficient ways to communicate reinforced the importance of such systems. Demand for "intelligent" and "human-like" applications has led researchers and developers in the field of Human Language Technologies to attain products such as *TRAINS* [1], *LIMSI ARISE* [2] and *Philips TABA* [3], in the train timetable information domain. Our system goes further than train scheduling and allows the user obtain a wide range of information about a trip (e.g., prices, duration, on-board services) while providing a relatively user-friendly mixed-initiative communication. In addition, this system has been developed for Spanish, a language with barely any such tools.

A very important issue in such systems, and a main concern for the current work, is poor speech recognition and its side-effects on the system modules to handle it. [4] claims to deal with them in the framework of unification grammar. Litman's [5] spoken dialogue system, *TOOT*, adopts a rather pragmatics-oriented approach: it automatically adapts its dialogue strategies so as to predict and respond to problematic input. [6], on the other hand, emphasises a more semantic-driven approach, as it is a more robust strategy. Our system follows this latter strategy and uses a semantic extractor that performs robust and flexible searches based on partial parsing, phrase structures, lexical key-words and morphological information (cf. section 4). Further, our system has also implemented a number of pragmatics-oriented strategies for the dialogue manager to overcome communication difficulties generated by poor input [7].

The starting point of our system was the study of real data (a human-human corpus and a human-machine, *Wizard-of-Oz* technique-based [8], dialogue corpus

collected) whose results were used for the development and training of the system modules and which represent a very valuable resource for further research as such in Spanish language.

The architecture of the system follows that of other such dialogue systems [7], [9]. The speech recogniser [10] makes use of an acoustic and a language model, the former being language-specific and the latter domain-specific and thus built for our task. As expected when dealing with spontaneous speech, the recogniser needs to incorporate the treatment of extra-linguistic phenomena. Yet, there remains a certain amount of noise and problematic input that will go on to the next module. The way the understanding module deals with these problems (cf. section 2), which makes the system more robust, is the main issue in this paper.

The understanding module starts with a linguistic processing of the transcription of the spoken utterance (cf. section 3) so as to extract its semantic content and formalise it inside *frames* that summarise the intervention (cf. section 5.1). These *frames* are sent to the dialogue manager [7] that also generates *frames* to transmit the necessary information to the natural language generator. Such *frames* are converted into sentences that the synthesiser outputs as a spoken response to the user's query.

2 Problems to be Tackled

The understanding module receives the transcription of the spoken utterance generated by the recogniser. Unfortunately, as it is well known in spontaneous speech, erroneous transcriptions take place. On the one hand, the recogniser is not perfect and it generates transcription errors itself. On the other, dealing with spontaneous speech implies having to face problems such as disfluencies.

Three different types or recognition errors must be considered: the first one is **excess of information**, i.e., the recogniser adds words that do not belong to the user's utterance. The second one is **erroneous recognition**, i.e, words detected by the recogniser do not match those really uttered by the user. The third error type is **grammar errors**, i.e., orthographic transcriptions can produce grammatical errors, including changes in grammatical categories.

These three error types can be handled at two different levels. At the recognition level, the tool should be tuned for the task domain. Also, the strategy of closing the entry channel, when necessary, should be considered. At a language processing level, our semantic information extractor can handle noisy input by focusing its extraction on specific syntactic phrases, lexical entries (word forms and lemmas), and even POS labels (cf. section 4).

Syntactic disfluencies have also been considered when designing the semantic information extractor. Most cases are sorted out at this stage (cf. section 4). Other disfluencies, such as lexical variations, pauses, noises,..., must be tackled by tuning the speech recogniser. Yet, the result of semantic extraction from the received input will also be confirmed by the dialogue manager when this is considered necessary for the query. This is a further attempt to get round any misrecognitions.

3 Tuning NLP Tools

This section describes briefly the tuning of the morphological analyser *MACO+* (*Morphological Analyzer Corpus Oriented*) [11] and the shallow parser *TACAT* [11] to our dialogue system. Both are robust and wide-coverage tools.

3.1 Morphological Analysis

The study of the corpora developed and the railway company DB has lead us to refine the task lexicon: it has provided us with all necessary city and station proper names (which have been labelled in a specific manner so as to ease their recognition), as well as guided us in the reduction of the lexicon to the most-frequent domain words. All tests carried out with the task lexicon have given the expected results, i.e., all forms have been assigned the correct label.

Besides reducing the number of forms, we have also reduced ambiguity in agreement with the specificity of the domain. *MACO+* assigns lemma/POS-tag pairs for each word form and then, the statistical tool *RELAX* (*Relaxation Labelling Based Tagger*) [11] disambiguates any remaining ambiguous words.

3.2 Syntactic Analyser

The syntactic analysis is performed with *TACAT*, a chart-based analyser making use of a chunk grammar that does not express any dependency relation between the constituents [11]. The grammar is context-free since it was originally designed for non-restricted domain language and later adapted for the present application. This adaptation has consisted in a) taking certain analysis rules out, such as those for verbal periphrases, and mostly b) adding domain-specific rules, in particular those to detect days of the week, dates, hours and minutes.

This analyser takes as input the output of the morphological tagger. Then it generates syntactic trees where the non-terminal nodes are syntactic labels such as NP, PP, VP and the terminal ones contain the word form, lemma and POS tag. These trees become the input for the semantic information extractor.

4 Information Extraction and the PRE+ Environment

The semantic information to be generated for the dialogue manager is based on the concept of *frame* [12], which can be easily translated into a DB query. A *frame* can convey two different types of information: *concepts*, which are the specific information the user is enquiring about, and *cases*, which are the restrictions to be applied to the *concepts*. For instance, *DepartureTime* is a *concept* which can have different *cases*, such as *DepartureCity*, *DestinationCity*, etc.

Information extraction is performed by means of the production rule environment *PRE+* [13]. *PRE+* works with rules whose model is: *condition* ==> *action*. *Conditions* establish the morphosyntactic context to be satisfied for *actions* to be applied. For instance, in Spanish, *DepartureCity* is always preceded

by prepositions *de, desde* (from). Thus, as expressed in this rule, any city name preceded by *de, desde* can (and must) be interpreted as the value for this *case*:

```
(rule DepartureCity1
  ruleset DepartureCity
  priority 10
  score [0,_,1,0]
  control forever
  ending Postrule
  (InputSentence ^tree <+a>tree_matching(
   [{pos=>grup-sp} [{lema=> de|desde}] [{pos=> np000c0, forma=>?forma}]]))
  (consulta ^attributes +attr)
   ->  (?_ := Print(DepartureCity,?forma))
       (?_ := REM(DepartureCity,X,+a))
       (+attrfi := NRPush(DepartureCity,?forma,+attr))
       (delete 2)
       (create consulta ^attributes +attrfi))
```

The main condition in this rule is the one starting with *InputSentence*. This condition specifies the type of syntactic element, *grup-sp* (PP), to be searched for by the rule. This element must be a node of the syntactic tree and must have two daughters: the first one must have one of the prepositions *de, desde* as lemma; its sister must be a city proper name (must have tag *np000c0*[1]). Since this is the name providing the necessary information, *DepartureCity*, this form must be kept (as indicated in *forma=>?forma*) to be later used in the action of the rule.

The last five lines of the previous rule express the actions of the rule. Leaving aside internal elements related to the functioning of *PRE+*, the third line in this block is the main action, which extracts the information and creates the *case+value* that will be sent to the dialogue manager in the next *frame*.

The main advantage of this formalism is that it allows to search for semantic information in a very flexible variety of ways: searching for word forms, lemmas and POS tags. Regarding syntax, phrase structure can be specified as much as required, if relevant, but it can be left unspecified otherwise. This makes the system considerably robust when facing problematic input received from recognition. For instance, the condition can be more restricted to specify either the type of sister nodes our initial *grup-sp* must have, or how many daughter (and/or granddaughter) nodes it can have, etc.

Below follows a rule that holds a more complex condition. In this case, the specification of syntactic structure starts at sentence level (*S*). *S* must have two daughters, a PP and an NP. The daughters of the PP must be exactly *con, salida* (with, departure), while its sister NP must have as nucleus a city name:

```
[{pos=>S}
  [{pos=>grup-sp} [{lema=> con}] [{lema=> salida}]]
  [{pos=>sn} [{pos=> np000c0, forma=>?forma}]]]))
```

[1] Digit #6, for semantic case, contains **c**, meaning *city* name (while **e** means *station*).

4.1 Rule Organization

Rules are structured according to a hierarchy that is established *a priori* and that determines their application conditions. The mother ruleset is called **top** and its daughters are mostly domain-specific **rulesets**. The dependecy relation between rulesets is always expressed with an *isa* relation. Each defined **rule** belongs to a unique ruleset, relation expressed with the *ruleset* property. Given these relations, rules inherit their properties from the rulesets they belong to.

4.2 Ruleset and Rule Control

Rulesets have several control properties associated which direct the way to apply rulesets. There is a global (top-level) control at an inter-ruleset level that manages the order of ruleset application and allows to handle complex or ambiguous situations, such as, potential rule overlapping at an inter-ruleset level.

The strategy of ruleset application has been established following the studied corpora. The first rule applied when the semantic information extractor starts is rule **top1** (shown below), which belongs to ruleset **top** and it has high priority. The action of this rule creates an ordered *concept* list that determines the order of evaluation for the rulesets. This is due to the fact that each *concept* is associated to one ruleset.

```
(rule top1
  ruleset top
  priority 1
  control one
  ->  (create concepts ^list
    [CS_DepartureTime
    DepartureCity, DestinationCity, DepartureDestinationCity, [...], Fi])
   (create consulta ^attributes [])
   (?_ := Tacat2WM()))
```

At a rule level, there are also some control properties. The initial part of the left-hand side of one *DepartureCity* rule is shown here:

```
(rule DepartureCity1
   ruleset DepartureCity
   priority 10
   score [0,_,1,0]
   control forever
   ending Postrule
   (InputSentence ...
```

The priority of a rule inside the ruleset is expressed by means of the **priority** property. Despite the fact that the **control** property is already established at a ruleset level, it is also possible to restrict the application of individual rules using the same property.

The most important rule property is that of **score**. Combining the values of its four parameters a large number of syntactic analysis problems can be tackled,

problems that are due to the use of spontaneous speech (such as the example below) or misrecognition (cf. section 2). These parameters permit to express a) the use of tree permutations, b) the maximum depth of the search level, c) the maximum number of leaves allowed in between terminal elements, and d) the maximum number of leaves allowed inside each element. For example, parameter (c) will have value 0 in user turns like: "Can I have the timetables for the Barcelona Bilbao trains?", allowing us to extract the *DepartureDestinationCity* value, despite not having the usual prepositions ("*from* Barcelona *to* Bilbao") next to the city names.

5 Semantic and Pragmatic Frame Tuning

Although our system is based on a detailed study of the domain-specific corpora, there are pragmatic issues that need to be taken into account. *Frames* have been tuned to the needs of the information exchange protocols and the limitations of the modules in order to achieve a more robust communication between the user and the system, as well as between the modules themselves. This tuning has helped simplify the existing structure and has been carried out bearing in mind the speech acts considered for this particular task. On the one hand, we have the **User Speech Acts** (*Query*, *Incomplete*, *Return Trip*, *Affirmation*, *Negation* and *Closing*) which have been resumed to four different frames (cf. section 5.1). On the other, there are the **System Speech Acts** (*Query to the User*, *Explicit Confirmation*, *Implicit Confirmation*, *Answer to Query*, *Lack of Understanding*, *Opening*, *Closing*, *Offer* and *Web Stalled*) which have been summarised in seven different *frames*. Thus, the system only needs to handle 11 main *frames*, which contain a very rich variety of information in a very limited number of structures.

5.1 *Frames* in the Understanding Module

To start with, given that semantic extraction is performed by means of a set of rules written in *PRE+* (cf. section 4) and that this environment is based on lexico-syntactic patterns, all reasoning behind the content of a *frame* and its corresponding labelling is carried out by the dialogue manager. This entails some implementation constraints for the understanding module:

- This module will only generate a set of *frames* carrying one of the following: a) precise queries to the system; b) *case* information that cannot be allocated by this module in any *frame* and that can only be tagged as *incomplete*, and c) answers to system queries that only hold an affirmation or negation.
- *Query frames* are marked with **Q** and those containing incomplete information with **I**. *Affirmation* and *negation* are labelled **A** and **N**[2], respectively.

[2] *Closing* is covered by *Negation* since the former is always preceeded by a negation and the dialogue manager is aware of the type of question such negation replies to.

– *Frames* will contain either a generic label or a non-generic one, where the former has one information field and only refers to the type of *frame*. The latter allows to include further information fields and thus provide further specifications on a particular *frame*. For instance, **I** opens a generic-label *frame*, while **Q** initiates a non-generic one. A *query frame*[3] does always specify what particular information the user is enquiring about:

```
((Q_DepartureTime
     (DepartureCity Barcelona)
     (DestinationCity Bilbao)))
```

while a generic *incomplete frame* providing information about two attributes for a query looks as follows:

```
((I
  (DepartureDate 10-05-2001)
  (DepartureInterval 13.00-21.00)))
```

The complete *frame* is encapsulated as an information pack by means of brackets. There are three different levels of brackets: a first one containing each *case/attribute* and its value; a second one enclosing all *cases* with their *frame* label, thus building up simple *frames*, and a third one enclosing all information relevant to that dialogue turn. The latter can contain a single *frame* or allow for the building of *frame sets*, as it happens in the query below:

```
((Q_DepartureTime
     (DepartureCity Barcelona)
     (DestinationCity Bilbao))
 (Q_Price
     (DepartureCity Barcelona)
     (DestinationCity Bilbao)))
```

This frame encapsulation makes the communication exchange easier for the awaiting modules since it establishes the whole information structure and thus informs the coming module whether it should be waiting for any further simple *frames* within a structure or not.

6 Conclusions

This paper describes the strategies adopted to handle problematic input and ease communication between modules in a Spanish railway information dialogue system for spontaneous speech. The strategies here presented are those concerning

[3] The list of query labels considered for the current domain task is: *Q_DepartureTime*, *Q_DepartureTime-R*, *Q_ArrivalTime*, *Q_ArrivalTime-R*, *Q_Price*, *Q_DepartureStation*, *Q_DestinationStation*, *Q_TripDuration*, *Q_TrainType*, *Q_Services*, *Q_DepartureDate*, *Q_DepartureDate-R*, *Q_ArrivalDate* and *Q_ArrivalDate-R*. Labels ending in **R** refer to the return trip, avoiding to create a specific *frame* for this speech act.

the understanding module and have allowed the development of a considerably robust system.

On the one hand, the tuning of NLP tools has helped reduce morphosyntactic ambiguity, thus lending a hand to our semantic information extraction environment, *PRE+*. This environment is highly flexible, allowing to perform searches as restricted or general as necessary, and by means of syntactic structures, lexical items and morphological elements (such as lemmas or POS tags).

On the other, the semantic *frames* have been designed so as to avoid ambiguity and store semantic information in a clear and easy-to-exchange manner.

Finally, both system and speech corpora developed constitute important resources for future research in Spanish language.

References

1. Allen, J.F., Miller, B.W., Ringger, E.K., Sikorski, T.: A Robust System for Natural Spoken Dialogue. *Proc. of ACL'96* (1996)
2. Lamel, L., Rosset, S., Gauvain, J.L., Bennacef, S. The Limsi Arise System for Train Travel Information. *Proc. of ICASSP'99* (1999)
3. Souvignier, V., Kellner, A., Rueber, B., Schramm, H., Seide, F. The Thoughtful Elephant: Strategies for Spoken Dialogue Systems. *IEEE Trans. Speech Audio Processing* **8** (2000)
4. Nakano, M., Miyazaki, N., Yasuda, N., Sugiyama, A., Hirasawa, J., Dohsaka, K., Aikawa, K. WIT: A Toolkit for Building Robust and Real-Time Spoken Spoken Dialogue Systems. *Proc. of 1st SIGdial Workshop on Discourse and Dialogue*, Hong Kong (2000)
5. Litman, D.J., Pan, S. Predicting and Adapting to Poor Speech Recognition in a Spoken Dialogue System. *Proc. of 7th National Conference on AI (AAAI-2000)*, Austin, Texas (2000)
6. Zue, V.W., Glass, J.R. Conversational Interfaces: Advances and Challenges. *Proc. of the IEEE. Special Issue on Spoken Language Processing* **88(8)** (2000)
7. Álvarez, J., Arranz, V., Castell, N., Civit M. Linguistic and Logical Tools for an Advanced Interactive Speech System in Spanish. *Engineering of Intelligent Systems (LNAI 2070)*, Springer Verlag (2001)
8. Fraser, N.M., Gilbert, G.N. Simulating Speech Systems. *Computer Speech and Language* **5(1)** (1991)
9. Giachin, E., McGlashan, S. Spoken Language Dialogue Systems. *Corpus-Based Methods in Language and Speech Processing*, ed. by Young, S. and Bloothooft, G., Kluwer Academic Publishers, Dordrecht (1997)
10. Bonafonte, A., Mariño, J.B., Nogueiras, A., Rodríguez, J.A. RAMSES: El Sistema de Reconocimiento del Habla Continua y Gran Vocabulario Desarrollado por la UPC. *Proc. of VIII Jornadas de I+D en Telecomunicaciones*, Madrid (1998)
11. http://www.lsi.upc.es/~nlp/tools.html
12. Minker, W., Bennacef, S., Gauvain, J.L. A Stochastic Case Frame for Natural Language Understanding. *Proc. of ICSLP'97* (1997)
13. Turmo, J. PRE+: A Production Rule Environment. Tech. report LSI-99-5-T (1999)