# CONTROLLING A HIFI WITH A CONTINUOUS SPEECH UNDERSTANDING SYSTEM

*J. Ferreiros, J. Colás, J. Macías-Guarasa, A. Ruiz, J. M. Pardo*

Grupo de Tecnología del Habla - Departamento de Ingeniería Electrónica
E.T.S.I. Telecomunicación – Universidad Politécnica de Madrid
Ciudad Universitaria s/n, 28040 Madrid Spain

## ABSTRACT

In this paper we present a speech understanding system that accepts continuous speech sentences as input to command a HIFI set. The string of words obtained from the recogniser is sent to the understanding system that tries to fill in a set of frames specifying the triplet (SUBSYSTEM, PARAMETER, VALUE). The understanding module follows the philosophy presented in [1]. The triplets are finally translated into infrared commands by an actuator module to be sent to the HIFI set, composed by a radio, a three deck CD player and a two tape cassette recorder/player. All circumstances (understanding incompleteness, HIFI set status, result of the command execution) are confirmed back to the user via a text to speech system with substitutable-concept pattern-based generated messages. We have introduced a response module because some of the final users will be blind people, and because we are studying the possibility of establishing restricted dialogues with the users in order to complete or correct the commands. The understanding engine is based on semantic-like tagging, including "*garbage*" tag, and context-dependent rules for meaning extraction. One of the system options allows the application developer to follow the reasoning process of the system (as every understanding rule has an associated concept pattern), spoken by the speech generation module. The concepts for speech generation are randomly substituted with alternative expressions having the same meaning to achieve a certain degree of naturalness in the response speech.

## 1. THE SPEECH RECOGNISER

The speech recogniser is based on a one-pass algorithm that searches for the best word sequence with a vocabulary of 163 words. The words are modelled as sequences of SCHMM allophone units [2,3]. A LSI board that includes a DSP ATT 32C performs the sampling and pre-processing stages. The rest of the recognition tasks are performed in the PC host. This configuration is working very close to real time.

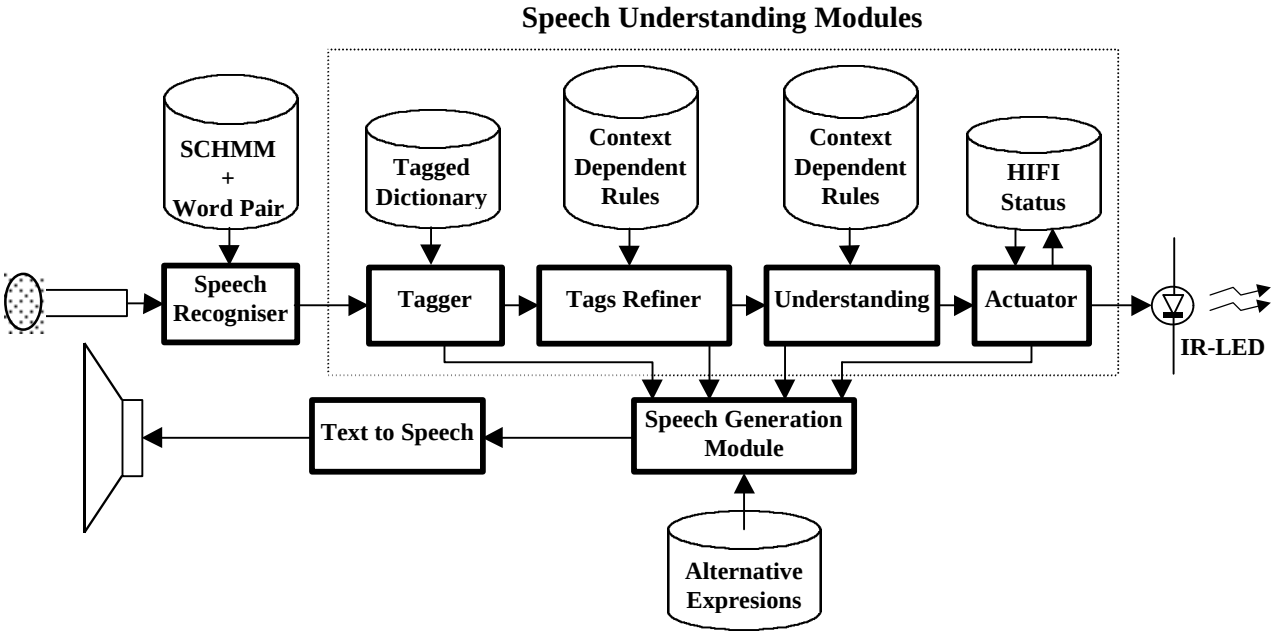


**Figure 1:** Module diagram of the whole system

## 2. THE SPEECH UNDERSTANDING ARCHITECTURE

The understanding architecture is composed of the following modules that run in the PC:

### 2.1. Tagger

This module receives the string of words recognised. Each word in the recognised string is assigned one or several semantic-like tags.

We have a set of 78 tags for this application. 18 of them are related to some specific action and correspond mainly to the verbs found in the command sentences (switch on, record...). 14 other tags are related to parameters and correspond mainly to system parameter nouns (volume, tone...). Other 14 tags are typically applied to values given to system parameters (increase value, am/fm...).

A "*garbage*" tag has been included so that it can be assigned when, for example, we are processing a function word that does not add any meaning in the semantic domain defined for the task. This method is a way of improving the robustness of the understanding system that has been designed to understand sentences without having to rely on the accurate recognition of function words, a hard task for the speech recogniser. It will be also useful in future developments where we will admit out-of-vocabulary words (using, for example, a parallel allophone grammar as model for unknown words or extending the recognition lexicon with more words than the ones specifically understood by the system). These words will also be tagged as "*garbage*" and will allow the correct understanding of sentences like "*please, set the volume higher*" where the word "*please*" may have not been considered in the design of the understanding system. There are a lot of sentences uttered by the users with unknown words that do not add extra meaning to the command. These sentences are correctly treated with this technique that improves, also in this sense, the robustness of the understanding system. Of course, if the unknown words that appear in a sentence are semantically relevant to the application, tagging them as "*garbage*" will not help to understand the sentence.

The set of semantic-like tags to be applied to each word is specified directly in the lexicon. For example, the word "*right*" has two possible tags: "*increment*" if the word is used as in the sentence "*move the volume knob to the right*" or "*position*" like in "*play the tape on the right*". The word "*program*" can be tagged "*programming action*" as a verb like in "program the set clock to one two zero zero", or can be tagged "*memory parameter*" as a substantive like in "*put program three of the radio*". One of the objectives of the following module will be to disambiguate them and choose one out of the set of tags a word can have or even change them by a new more specific one.

### 2.2. Tags Refiner

This module has the following aims: 1) numbers processing, 2) disambiguation of those words with several possible tags and 3) garbage removal. It receives the output of the tagger (the words with their corresponding sets of tags) and produces a similar, but more refined output. This output is the product of choosing or changing the tags or even a change in the literal expressions of the words. A change in the literal strings is exactly what happens in the first step of numbers processing. To ease the recognition system design, numbers composed of more than one figure are supposed to be uttered using only the figures between zero and nine. This implies a simple processing of the sentences to extract the right numbers. First, grapheme representations of the numbers are changed into the corresponding figures. Then, contiguous figures are grouped to form the corresponding single number. For example, the sentence piece "...two five..." is rewritten as "25". Garbage words may have served as separators to properly translate sequences of numbers. This is the reason why we remove garbage words just at the ending step of this tags refiner stage.

Next, the module applies a set of context dependent rules to disambiguate these words that have several possible tags and also to refine the tagging of some words if more information in the sentence is captured by the rules. These rules are expressed in terms of some functions that analyse the context searching for the existence of some specific tags. Both in this tags refiner stage and in the following understanding stage, we have two types of context searching functions: those that search throughout the whole sentence and others that search in the immediate context of the word of interest: the previous and following words. This latter type of functions give the system a proximity criterion to group words and refine with them the partial meaning of some phrases when necessary, while those functions that search everywhere give the system a flexibility in the phrasing order to construct the sentences.

Some examples of the disambiguation process are: for the word "*right*" we have a rule that states that "*if there exists any other word tagged as a tape parameter, then the word right is the position of this tape else it is a increment indicator*". For the word "*program*" we have a rule which intention is "*if the tag of the next significant word is a number, then the word program is used as the corresponding substantive else it is a programming action indicator*". Some other rules simply refine some tags when more information is observed. For example, if a word like "*disk*", tagged "*CD value*", precedes a word tagged "*number*", only the number is preserved and it is tagged as "*specific CD value*".

We have found that the order in which the rules are executed is of major importance to get good performance. As it will also be discussed for the next module, the understanding stage, we came to the conclusion that specific rules should be written and executed in the first place, leaving more general rules at the end.

Finally, we remove all garbage words. The output of this stage is a sequence of word-refined tag pairs, most of the times less in number compared to the input.

## 2.3. Understanding stage

Context dependent rules are applied to the output of the tags refining stage to extract the meaning of the sentence. The output of this stage is one or a set of triplets that specify the subsystem on which the action will be performed (radio, cd, cassette), the parameter we want to change (volume, tone, broadcast station memory number, song number, ...) and the value the parameter should be set to.

| |
|---|
| *There_is (tag)*: examines if the tag *tag* is present anywhere in the sentence. |
| *There_is_another (tag_list)*: examines if another tag, different from the ones present in the tags list *tag_list*, appear anywhere in the sentence. |
| *Previous_significant_tag (index)*: returns the tag of the previous significant word to the indexed word. It gives, with the help of the following function, a proximity criterion to the understanding procedure. |
| *Next_significant_tag (index)*: returns the tag of the next significant word to the indexed word. |
| *Word_tagged_as (tag)*: returns the literal of the word that has been tagged as *tag*. It is useful to obtain literals to fill in the corresponding slots of the frames. |
| *Index_of_Word_tagged_as (tag)*: returns the index of the literal that has been tagged as *tag*. |

**Table 1:** The set of basic functions, constituents of the context dependent rules

In the design of this first version of the understanding system we made the choice of trying to understand each sentence using only the strictly needed word-tag pairs delivered by the tags refining stage. We did not necessarily use all the pairs present after the tags refining stage to extract the meaning. For example, as soon as we see a literal whose tag is related to the volume parameter, we look for the needed extra information (basically in this case: if the volume should be elevated or decreased) and if it is found, execute the command. So, a sentence like "*set the volume of the cassette deck on the left higher*" will have the effect of simply elevating the only volume the HIFI set has. Maybe in this case the user is listening to the radio while the tape is playing and a message like "*you should first switch the sound source to the tape to be able to hear it*" is convenient for the user to reformulate the command. Also, both the user and the speech recogniser can generate nonsense sentences like "*switch the volume on*" that in this version will be executed trying to switch the set on, not examining the meaning of the word volume because no increase/decrease value can be found.

The context dependent rules of this module try to extract enough information from the sentence to fill in one or a set of (SUBSYSTEM,PARAMETER,VALUE) frames that can be directly executed by the actuator stage. These rules make use of the basic context analysis functions that we show in Table 1.

Following the design criterion of not necessarily examining all the input pairs, we consistently put more specific rules in the first position. The specific rules look for literals with very specific tags that are almost a command in its own (like a sentence that includes the word "*fm*" where the assumed interpretation, without looking at any other word, is "*set the radio in fm band*"). Many of the specific rules use directly the function *There_is (tag)* as the only condition to trigger an interpretation. Then we put rules that progressively need to examine more terms to determine the correct interpretation like in the sentence "*switch memory two on*". This second set of rules makes extensive use of the rest of the basic context analysis functions in their condition part. Finally we put more general rules that, with the knowledge that no previous specific rules have been able to produce an interpretation, relax the exigencies and produce a more general interpretation like for the sentence "*switch on*".

Most of the rules of this stage are only triggered if no previous rule has been triggered allowing the behaviour that more general rules run only if no more specific rules have been used. Nevertheless, there are some cases where a rule is written to interpret only a part of a sentence, being compatible with the rest of the rules trying to interpret the rest. For example, in the sentence "*Record the song number six of the disc one*", one rule will be involved with the playing of the specified song on the correct CD, another rule will switch the record sound source to CD and another rule will issue the record command to the only tape that can actually make the recording in this set. So, five different frames will be filled in by the rules and executed by the actuator. The first rule will produce three frames: (CD, DISK_SELECTION, 1), (CD, SONG_SELECTION, 6) and (CD, PLAY, void). The second rule will produce the frame (TAPE, RECORD_SOURCE, CD) and finally, the last rule will fill in the frame (TAPE,RECORD,void) where the tape does not need to be specified because only one of them can record. To allow such behaviour it is very important the order in which the rules are written (and consequently executed) taking into account the possible inhibition effect suffered by some rules.

Another characteristic of most of the rules employed is that they produce a correct interpretation independently of the order in which the constituent phrases of the sentences appear. This allows the understanding of sentences ranging from the prosaic "*Set the volume of the radio higher*" to the poetical "*In the radio the volume set higher*". It is not the specific sequence of concepts, but their conceptual contents and the order-indifferent relations among them, which leads to the correct interpretation of the command.

Each rule fills in two message strings, one to confirm the action that will be carried out or to inform about any understanding problem and another where the system "ex-

plains" its reasoning. The reasoning string is also filled in by the tagger and tags refiner stage. These message strings are composed of the concatenation of literal messages and concept marks to be processed by the speech generation module.

For example, a rule of the tags refining stage disambiguates the verbs that may have an increasing meaning if another word with this meaning is found. If the correct conditions are met, this rule fills in a string like: "*C_SEEING the word **W1** with an increment meaning, C_THINK that **W2** means an increasing action*", where in W1 and W2 the rule writes the specific words on which it is working. C_SEEING and C_THINK are two marks of substitutable concepts. As it can be seen this is a pattern based approach to speech generation.

Another example from the understanding stage: a rule examines if a word with a tag related to a set mode is found and directly issues the frame to change the set mode. It then fills in the following pattern: "*C_SEEING **W1**, I set this mode*"

## 2.4. Actuator

This program reads the triplets with the actions to be taken and execute them generating the right coded waveform on an infrared LED. It also keeps track of the HIFI set status, so that some actions are corrected. For instance, if the user ask the set to be switched on while the set is already on, the actuator will not issue this command again. The user is informed of these circumstances using the speech generation module.

# 3. THE SPEECH GENERATION MODULE

The input to this module are strings composed of literal messages and concept marks that are randomly substituted with expressions carrying the same meaning to achieve a degree of naturalness in the synthesised messages. In the example: "*C_SEEING the word higher with an increment meaning, C_THINK that put means an increasing action*", C_SEEING can be substituted by a set of expressions like: "*As I can see*", "*As I have discovered*", "*As It appears*", ... and C_THINK by expressions like: "*I think*", "*I imagine*", "*I suppose*"...

A VISHA board, designed in the Speech Technology Group, housing another DSP ATT 32C, hosts the speech synthesiser subsystem.

# 4. DISCUSSION AND FUTURE WORK

In this speech understanding application we have used semantic-like tagging of the words and context dependent rules to extract the meaning of commands uttered in continuous speech. The use of the tag "*garbage*" gives the system a certain degree of robustness against recognition errors for short function words. The kind of rules we are applying to refine the semantic tagging and to finally interpret the sentence is dependent on the context but not on the order of the words that represent this context (except for these phrases where the previous and following words are used to disambiguate the tag of another word), allowing flexibility when issuing the commands. The speech response subsystem is based on pattern based speech generation with the use of randomly substitutable concepts, which produces a naturalness sensation in the user.

We have to work more on the treatment of all information available in the sentence to detect inconsistencies that the user should be aware of. This work will also prevent the system to directly execute a command due to a recogniser error that includes in the string of words one with a direct interpretation rule.

We want also to handle correctly more complex commands. For this purpose we have used in other applications a more complex architecture that includes semantic-syntactic parsing to extract the sentence structure.

We have also realised that there are many points in the system where a dialog can be established with the user to complete or correct information. In this dialog points a speech query should be issued to the user and a recogniser restricted only to the expected answers (a word spotting algorithm for example) could be launched to obtain the data needed.

# 5. ACKNOWLEDGEMENTS

The authors wish to thank in general to all Speech Technology Group members for their help and specially to Natalia París Navajas and Yolanda López Moreno for their work on the speech recogniser for this system.

# 6. REFERENCES

1. José Colás, Juan M. Montero, Javier Ferreiros, José M. Pardo , "An alternative and flexible approach in robust information retrieval system", EUROSPEECH' 97, ISSN 1018-4074, Rhodes, Greece

2. Javier Ferreiros, José M. Pardo , "Preliminary Experimentation of Different Methods for Continuous Speech Recognition in Spanish", EUROSPEECH' 95, ISSN 1018-4074, Madrid, España

3. J. Ferreiros, "Contribution to Markov Models training methods for continuous speech recognition". Ph D. Thesis. Universidad Politécnica de Madrid, 1996