A Speech Agent for Remote E-Mail Access

S. Raptis†, C. Malliopoulos†, S. Bakamidis‡, and G. Stainhaouer‡

†National Technical University of Athens
Zographou Campus, GR-157 73, Athens, GREECE

‡Institute for Language and Speech Processing
Epidavrou & Artemidos St., GR-151 25, Athens, GREECE

e-mail: spy@ilsp.gr, Tel: +30-1-68 00 952, 6, 9, Fax: +30-1-68 54 270

ABSTRACT

This paper presents the design and implementation issues for a desktop agent running under the MS-Windows 95 environment. This agent is the result of the research and development efforts of the authors on speech processing at the Institute for Language and Speech Processing. It has emerged as an extension of the authors' previous work on speech interfaces for visually impaired persons and it is currently supporting only Modern Greek.

The Speech Agent (SA) is responsible for delivering e-mail messages through telephone lines employing speech synthesis and speech recognition techniques. A user can "connect" to the agent by simply placing a call to a phone connected to the PC. Using a standard predefined set of commands (words or sentences), the user can then instruct the agent to read an e-mail, check for new e-mails, delete e-mails, etc.

THE SYSTEM

The Overall System Architecture

The system roughly consists of a speech synthesis module and speech recognition module. The system layout is shown in Figure 1.

System Requirements

Apart from the MS-Windows 95 environment and a standard compatible sound card, SA software only requires a telephony card, keeping the system at low cost. A moderate PC system, e.g. a Pentium at 233MHz with 32MB of RAM or better, is sufficient for the system to execute close to real-time.

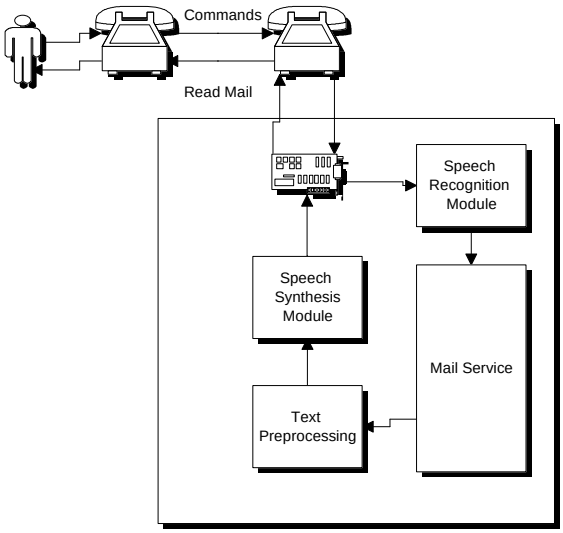

Fig. 1 – The overall system architecture

THE SPEECH SYNTHESIS MODULE

The speech synthesis module is based on the ILSPeak engine developed at ILSP for Modern Greek. ILSPeak is based on the conventional Klatt's cascade/parallel speech formant synthesizer [1] where some modifications have been made both to the excitation sources and the vocal tract filter specifications. Some of the modifications were motivated by the fact that output speech quality and overall system performance can be substantially improved with the utilization of more accurate and computationally efficient models of the voicing source and vocal tract, while others were dictated by the articulatory structure of the language at hand.

A block diagram of the synthesizer is given in Figure 2.

The system's segmental rulebase currently contains about 100 rules, a number which is expected to further increase in future.

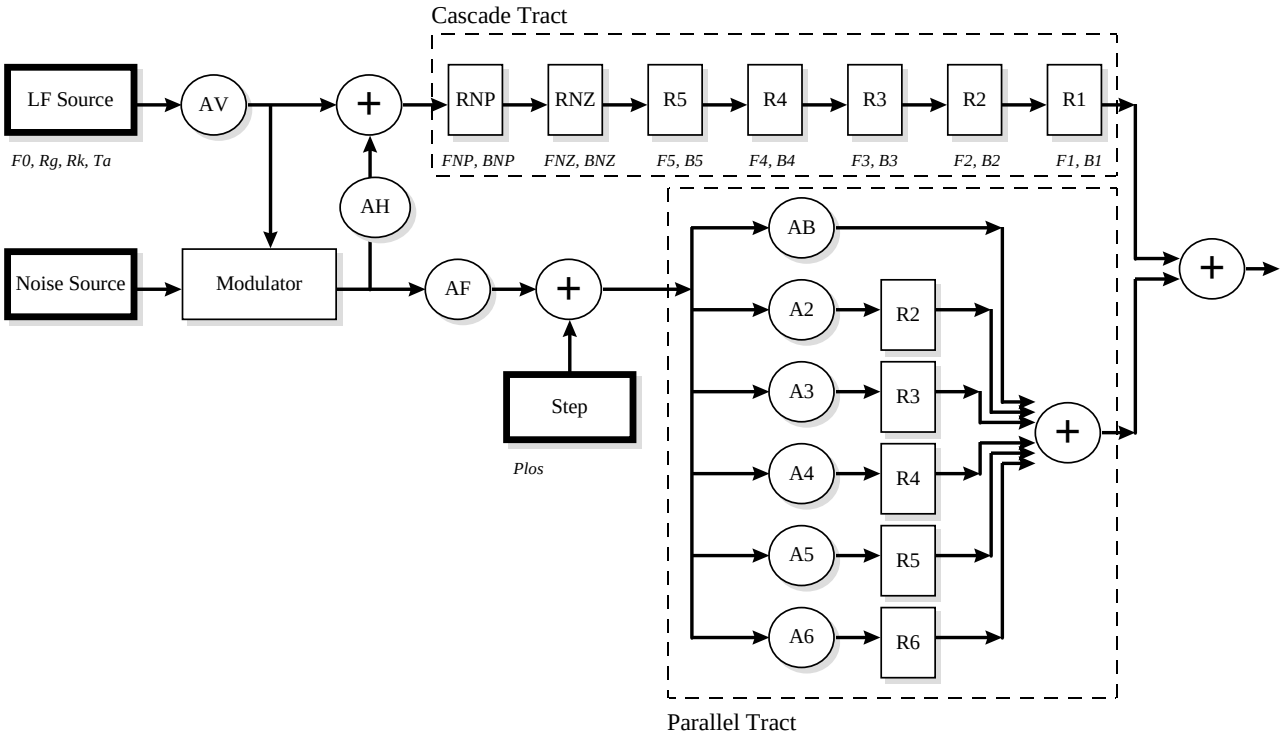

Fig. 2 – The speech synthesis module

THE SPEECH RECOGNITION MODULE

The speech recognition module can recognize a predefined set of commands (words or sentences). This module is based on a connected speech recognition system, shown in Figure 3, where in the analysis phase the cepstrum is extracted.

The Recognition Procedure

In particular, the spectral analysis uses an LPC front end with the following characteristics:

- anti-aliasing filter with a cuttof frequency of 4 KHz
- sampling frequency of 8 KHz
- analysis window of 256 samples (32 msec) with a shift a 100 samples (12,5 msec)
- cepstrum order equal to 10

The one-stage dynamic programming algorithm for connected word recognition [2] is used in the comparison stage of the recognition module. In the training phase 20 users (10 men and 10 women) uttered the commands five times. In order to reduce the computational complexity of the recognition module a clustering technique is used in the learning phase. The most representative reference patterns for each word has been extracted by using the modified *k*-means algorithm

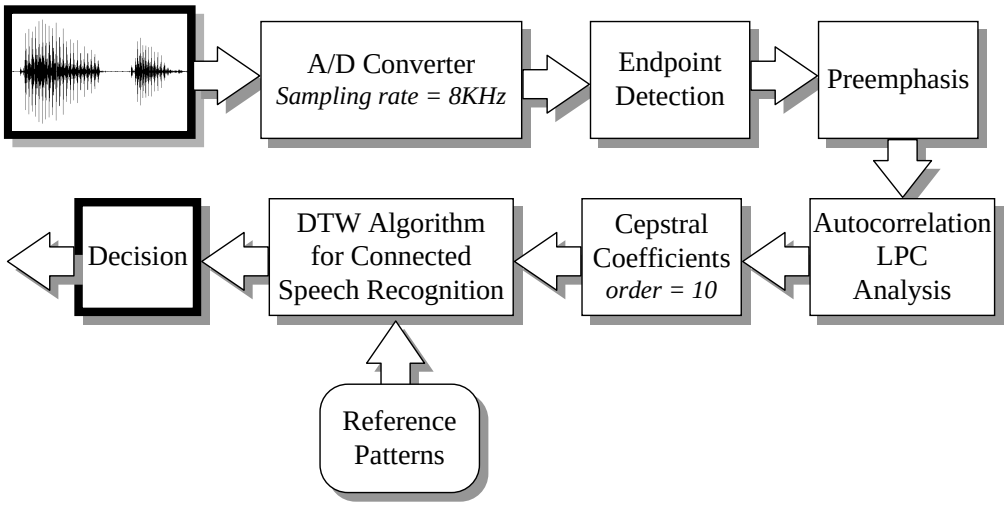

Fig. 3 – The speech recognition module

[3]. The pattern matching algorithm used in the clustering procedure is the Dynamic Time Warping Algorithm as proposed in [4] with weighted cepstral distance.

A fact to stress concerns the patterns used for training the speech recognition module. The patterns used

The Command Set

At each time, one message is considered to be the current message and the commands specified are used to manipulate it.

The module recognizes the following commands:

- **`Read message`**: Read the full contents of the current message.
- **`Delete message`**: Removes the current message.
- **`Stop`**: Stops playback of the current message.
- **`Previous message`**: Sets the previous message as the current message.
- **`Next message`**: Sets the next message as the current message.
- **`First message`**: Sets the first of the new messages as the current message.
- **`Last message`**: Sets the last of the new messages as the current message.

Apart from these commands, the recognition module offers to the user the possibility to store and handle previously recorded aliases for commonly used e-mail addresses. For example, assuming that we have a coworker named John Taylor we may utter the name "**`Taylor`**" a few times, train the system using these utterances, and add it to the alias list of the system connecting it to the e-mail address of John Taylor, e.g. **`J.Taylor@ilsp.gr`**. Then, we may issue through the telephone a command of the form:

`Forward to <`*`alias`*`>`

where **`<`*`alias`*`>`** might be **`Taylor`**.

An additional ability that the user is given, is to switch from a word/sentence recognition mode to a spelling mode and vice versa by issuing the following commands:

- **`Spell mode`**: Switches from word/sentence recognition mode to spelling mode, i.e. recognition of letters and digits.
- **`Command mode`**: Switches from spelling mode to word/sentence recognition mode.

In the spelling mode, the user can enter any combination of letters, digits and symbol characters (e.g. "**`@`**" as "**`at`**") and can thus spell an arbitrary e-mail address to forward the current message to.

To maximize the distance of the letter patterns, we used the typical spoken alphabet shown in the following table.

Table 1 – The spoken alphabet

A	Alpha	B	Bravo	C	Charlie
D	Delta	E	Echo	F	Foxtrot
G	Golf	H	Hotel	I	India
J	Juliette	K	Kilo	L	Lima
M	Mike	N	November	O	Oscar
P	Papa	Q	Quebec	R	Romeo
S	Sierra	T	Tango	U	Uniform
V	Victor	W	Whiskey	X	X-ray
Y	Yankee	Z	Zulu		

We have not included the ability to compose a new message dictated in spelling mode, since this process seemed rather inefficient and time consuming for the user.

THE TEXT PREPROCESSOR

The SA contains a text preprocessor which enables it to perform tasks as:

- skipping mail headers and other unwanted information
- correctly expanding numbers, dates, etc.
- expanding abbreviations (based on a stored association table)
- expanding aliases, etc.

Special symbols such as asterisks around a word are interpreted as emphasis markers to enhance the naturalness of the synthesized speech.

OPERATIONAL ISSUES

The sequence of events when retrieving e-mail messages through the telephone, is as follows:

1. The user calls the PC from a telephone.
2. As soon as the ring is detected, the telephony card "picks up" the phone, retrieves the new messages from the mail server, and reports its status in the form:

   "**`You have <`*`num`*`> new messages.`**"

   where **`<`*`num`*`>`** is the number of new messages retrieved or "**`no`**" if none was found.
3. The first new message is set as the current message.
4. The *Subject* field of the current message is synthesized and played back through the telephone card to the user.
5. The recognition module waits for appropriate commands and serves as they arrive by issuing

appropriate commands to the mail server and the synthesizer module. When **`Read message`** is selected, the full contents of the current message begin to get synthesized played back on a sentence-by-sentence basis till the message finishes or the **`Stop`** command is issued. Appropriate commands change the current message and the program loops to step 4.

6. If the user hangs up, the system is reset and is ready for the next call.

MAIL SERVICES AND OTHER "STANDARDS"

A viable speech interface should rely on widely available resources and standards as much as possible. On a PC under the MS-Windows environment, some "standards" are enforced by the operating system. It is of critical importance for rapid and maintainable development that these are adopted.

In this context, a significant effort is being made for the SA system to conform to existing standards and application program interfaces (API's). Such API's currently supported by Microsoft include:

- *The Telephony Application Programming Interface* (TAPI): It is used for the basic telephonic functionality, i.e. to control the telephony sound card input and output data flow and phone line handling.
- *The Messaging Application Programming Interface* (MAPI): It is used for mailing services. NETSCAPE API can provide an alternative.

CONCLUSIONS - FUTURE WORK

Most of the modules of the system have already been independently evaluated as stand alone applications or as parts of more sophisticated interfaces and/or products. Specifically the speech synthesis module, is scheduled for evaluation within the next months along with a speech-enabled text editing environment with the cooperation of the Greek Association of Visually Impaired Persons, and some first feedback will be available very soon.

Future work will concentrate on new emerging software technologies such as the Speech Synthesis and Speech Recognition SDK's of MICROSOFT which provide a common base for speech enabled software and are expected to play a dominant role in future user interfaces.

It is also possible to extend the system so as to provide the ability for multiple users with different accounts.

REFERENCES

[1] D. H. Klatt, "Software for a Cascade/Parallel Formant Synthesizer," *Journal of the Acoustical Society of America*, Vol. 67, No. 3, 1980.

[2] H. Ney, "The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-32, pp. 263-271, April 1984.

[3] J. G. Wilpon and L. R. Rabiner, "A Modified K-means clustering algorithm for use in isolated word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-33, pp. 587-594, June 1985.

[4] H. Sakoe and S. Chiba, "Dynamic programming optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26, pp. 43-49, Feb 1978.