A TALKING WORD PROCESESOR

David Hawthorn* and Chris White*

*Department of Business Systems
Monash University

ABSTRACT - Speech is becoming an increasingly common form of computer output. Its applications are any situation where the reading of a computer screen may be difficult. This paper looks at one very easy method of implementing speech in Microsoft Word for Windows (1).

INTRODUCTION

Speech is becoming an increasingly common form of computer output. Although computer speech output has many applications, the main interest of the authors is with print handicapped computer users. Unfortunately computer speech hardware and software often have a price tag which is not compatible with the majority of these users.

This paper presents a simple method of computer speech output in Word for Windows which is based on a macro by Edwards (1995). This method will work with any Windows application that has a macro language and supports Dynamic Data Exchange (DDE). If this macro is used in applications other than Word, some modifications may be necessary as not all Windows applications have the same macro language.

The following hardware and software are required for the method presented:

A 386 or better PC
SoundBlaster sound card (8bit or better) (2)
Microsoft Windows (3)
Monolog for Windows (often bundled with SoundBlaster sound cards) (4)
Word for Windows

With the popularity of home computers and sound cards for computer games many users would already have the majority of these requirements. Not only is this method simple, it is also inexpensive. The popularity of Word as a word processor and the SoundBlaster card for sound were important factors in deciding to use them.

Text-to-speech conversion is the method used for the speech production. This method does not require a large database of words (in text form) and associated sound files. The sounds produced are derived from the spelling of the words by using standard phonetic rules for pronunciation. There is the facility to add words to a library which allows the standard rules to be ignored. This library contains a list of words along with a phonetic description of each word. No sounds are actually stored. Words such as "read" are an obvious problem; they have two distinct pronunciations. Monolog pronounces the past tense form the same as the present/future tense. In order to resolve this, the grammatical structure of the sentence and the context may need to be analysed. Text-to-speech conversion of this quality is obviously complex and expensive. Intonation is also important in the meaning of speech. This would also be a feature of a more advanced text-to-speech conversion package.

The speech obtained from Monolog tends to be a little mechanical. This can be a problem as the listener can tire easily of a mechanical monotone voice. It is the common cost/quality trade-off. Better quality can be obtained by spending more money, but it may be more money than what many can afford.

THE MACRO

The following Word for Windows 6 macro will speak an entire Word document. The document is automatically scrolled so that the text being spoken is always in view. The sentence currently being spoken is highlighted.

The macro relies on a Dynamic Data Exchange (DDE) link between Word and Monolog. The current sentence is sent to Monolog and then the macro immediately (without waiting for the sentence to be spoken) proceeds to the next command. This forces a busy wait loop to be written to allow Monolog time to speak the current sentence. If this is not done the highlighting gets one step ahead of the speech. The highlighting cannot get more than one step ahead as the DDEPoke command will not be executed whilst the destination program (in this case Monolog) is busy.

```
           'loop to allow Monolog time to speak the sentence
           lpc = 100 * length
   lp:
           lpc = lpc - 1
           If lpc > 0 Then Goto lp
```

As can be seen, even with the attempt to link the loop count with the length of the current sentence, this could not be considered an elegant program. Apart from the obvious problem of the loop execution time being dependent on the computer's speed, the length of the sentence is not be directly related to the amount of time taken to speak the sentence. This is most easily seen at the end of paragraphs and in sentences which contain numbers. In a future working paper we will look at an alternative to DDE which solves this problem.

The following code shows the details of the macro. The actual speaking takes little code. Most of the code is to move consecutively through the sentences, highlight the current sentence and to restore it to the standard colour.

```
           Sub MAIN
           On Error Goto e1

   init:
           'initiate DDE
           ChanNum = DDEInitiate("Monolog", "talk")

           'get current position
           InitSelStart = GetSelStartPos()
           InitSelEnd = GetSelEndPos()

           'move to start of document
           StartOfDocument
           Goto b300

   nextsentence:
           another = SentLeft(1, 0)
           'loop to allow Monolog time to speak the sentence
           lpc = 100 * length
   lp:
           lpc = lpc - 1
           If lpc > 0 Then Goto lp
           SelectCurSentence
           CharColor 1  'black

           'select sentence
           another = SentRight(1, 0)
```

```
b300:
        a$ = "select sentence"
        SelectCurSentence
        all$ = Selection$()
        length = Len(all$)
        CharColor 9  'blue

        'do the actual talking
        a$ = "talk"
        DDEPoke(ChanNum, "Talk", all$)

        'goto next sentence
        a$ = "next sentence"
        another = SentRight(1, 0)

        'end of document
        a$ = "end of document"
        endoc = AtEndOfDocument()
        If endoc = - 1 Then Goto finish

        If another = - 1 Then Goto nextsentence
        On Error Goto 0
        Goto finish

e1:
        On Error Goto 0
        'errors come here
        MsgBox a$

finish:
        another = SentLeft(1, 0)
        SelectCurSentence
        CharColor 1
        'go back to initial position
        SetSelRange(InitSelStart, InitSelEnd)
        'close DDE7
        DDETerminate(ChanNum)
        End Sub
```

One tip which may be useful in large documents: if you want to stop the macro before it has finished speaking the whole document you can press Esc. The Macro should stop at the end of the next sentence.

CONCLUSION

The macro presented in this paper gives a method for speech production in a word processor. The speech is not of the highest quality but the method is simple and inexpensive. It has the potential to be of great benefit to the print handicapped.

ACKNOWLEDGMENTS

This work is part of development under the "Essex Project" support for which has been provided by Aridyne Corporation and Advanced Micro Systems.

NOTES

(1) Microsoft Word for Windows is a trademark of Microsoft Corporation.
(2) SoundBlaster is a trademark of Creative Labs.
(3) Microsoft Windows is a trademark of Microsoft Corporation.
(4) Monolog for Windows is a trademark of Creative Labs.

REFERENCES

Edwards, J.D. (1995) *Spoken Word* (in *Technical Tips*), Australian Personal Computer, 16:2, 190.

DEVELOPMENT OF A VERY FAST PREPROCESSOR

Young-Mok Ahn, Hoi-Rin Kim

Spoken Language Processing Section
Electronics and Telecommunications Research Institute

E-mail : aym@zenith.etri.re.kr

ABSTRACT — This paper proposes a very fast preprocessor for a large vocabulary isolated word recognition. This preprocessor extracts a few candidate words using the frequency and the time information for each word. For designing reference pattern, we use the order of amplitude of speech feature. So, the proposed preprocessor has a small computational load after the extraction of speech feature. In order to show the effectiveness of the proposed preprocessor, we compared it to a speech recognition system based on semi-continuous hidden Markov model and a VQ-based preprocessor by computing their recognition performances of a speaker independent isolated word recognition. In experiments, we use three types of speech database. The first, a speech database consists of 244 words including digits, English alphabets, etc. The second, a speech database consists of 22 words including section names. The third a speech database consists of 35 words. This preprocessor is composed of three major parts: the feature extraction, the feature sorting, and the reference pattern matching with reference templates. After sorting it requires only one vector addition per frame, namely, *vocabulary size x length of incoming frame*. In consequence, this approach is therefore much faster than our previous version which is the VQ-based preprocessor for isolated word recognition task (Ahn et al, 1994). In the experimental results, the accuracy of feature sorting based preprocessor is 99.86 % with 90 % reduction rate for the speech database of 244 words.

INTRODUCTION

In a large vocabulary task, the speech recognition system should have some efficient algorithms for reducing the computational load. To develop an efficient algorithm which is reducing the search space such as Viterbi beam search, it is necessary that the algorithm should not degrade the system performance. In real applications, the speech recognition system should have another efficient algorithm which can reduce the memory size. If the speech recognition system have a large memory space, it is not a emulous product. Considering the computational load and the memory size, the proposed approach shows the solvability on the isolated word recognition problem.

Although the proposed preprocessor is less exact compared to other speech recognition systems in the recognition performance of the first candidate word, it keep an appropriable preprocessor, because the preprocessor can extract a small candidate words including correct word from the large vocabulary within correct word on maintaining the system performance. And the preprocessor is easily constructed from the training speech database. On the other hand, use of the preprocessor for continuous speech recognition is not easy. This is because, to achieve the preprocessor with high accuracy performance, a large speech database with the precise labeling information should be prepared. So, the preprocessor combined in the continuous speech recognition system. To measure the performance of the proposed preprocessor, we performed the experiments on the speaker independent isolated word speech recognition. And we compared that with other speech recognition system performances.

In the rest of the paper, an overview of the proposed preprocessor and other speech recognition system is first introduced in the section of system overview, and than the performance of each system is described in the section of experiments. The experimental results and concluding remarks are given in the section of conclusions.

SYSTEM OVERVIEW

In order to show the effectiveness of the proposed preprocessor, we compared it to three types of speech recognition system for their recognition performances of a speaker independent isolated word recognition. The brief introduction is as below. Although the main purpose of each system is not same, but they provide a appraisal experimental result for isolated word recognition.

Feature sorting based preprocessor(FSP)

Recently, auditory system has begun to play a larger role in motivating the design of some speech recognition front-end systems. And spectral transitions play an important role in human auditory perception. In other words, the current speech signal is much influenced by the previous speech signals. And it is also meaningful to hypothesize that the speech signals have the relationship among orders of feature vectors. The idea of the feature sorting based preprocessor is derived from the hypothesis.

This preprocessor is composed of three major parts: the feature extraction, the feature sorting, and the reference pattern matching. After feature sorting it requires only one vector addition operation per frame, namely, *vocabulary size x length of incoming frames*. And the feature sorting requires very small computational load compared with vector quantization. The feature sorting based preprocessor is very similar to the VQ-based preprocessor except the vector quantization. Let $y_1, y_2, \ldots, y_n$ be a label sequence produced by the feature sorting based acoustic processor in response to an utterance of some unknown words. In the feature sorting based preprocessor of obtaining a short list of words, we seek a word scoring function of the form where S_w

$$S_w = \sum_{t=1}^{n} v(y_t, w) + i_w \qquad (w = 1, 2, \ldots, N)$$

denotes the score for word w, $v(y_t, w)$ denotes a real-valued vote cast by label y_t for word w, i_w denotes the initial value of S_w, and N denotes the number of words in the vocabulary. A short list can then be constructed from the highest scoring words.(Lalit et al, 1988) Further information for constructing reference patterns of the feature sorting based preprocessor can be found in (Ahn et al, 1994), (Lalit et al, 1988).