A CONNECTED SPEECH PARSE FOR AUSTRALIAN ENGLISH UTILIZING MATRIX SYLLABLE FORMATION

R.A. Bennett, E. Lai, and Y. Attikiouzel
Department of Electrical and Electronic Engineering
The University of Western Australia

ABSTRACT - A system is proposed to enhance the speed for Connected Speech Recognition systems by the formation of syllables from a phoneme string. Binary matrices are used to provide fast calculation of syllables which are used in modified dictionary search patterns. The system is designed for use with simple recognition systems which provide minimal allophonic information. Some preliminary results are discussed.

INTRODUCTION

Progress in the field of Connected Speech Recognition (CSR) systems is inhibited by the problems of locating word boundaries in a speech waveform. Various hardware/software systems exist for translating a speech waveform into a string of phonemes plus allophonic features, the degree of information depending on the sophistication of the system involved. The phoneme string is converted through a dictionary search algorithm to recover the most likely words uttered. For complex systems, this final stage uses the allophonic constraints to enable words to be efficiently located, but for more primitive systems (e.g. smaller, cheaper, and/or portable systems) the information contained in the phoneme string is minimal and the search routine may be time consuming, especially when the dictionary is large enough to require storage on disk.

It is for these simple recognisers that this system of syllable formation is proposed. Utilizing only the basic phoneme string with no higher allophonic information, and assuming no errors in the phoneme string, the system uses preset grammer rules to form one or more syllable strings from the phoneme string. These syllable strings are then matched in a previously compiled syllable dictionary in the same way as for phonemes strings. The advantage of this method is the increased length of a basic syllable unit

(approx 2-3 phonemes on average) which decreases the search time required for each string. If this saving in time is greater than the extra time required to form the syllables and the time spent checking alternative syllable possibilities, then an overall gain in speed is achieved.

The system was implemented as a C language software program, on a Apricot VX-150 running XENIX/386 with 2MB of RAM and the Macquarie dictionary (words and phonemic representations only) [2] stored on hard disk.

IMPLEMENTATION

Syllable Formation

The method used to create syllables in this system was previously used by Kenneth Church [1] as part of a larger parsing algorithm involving allphonic constraints. The method involves the use of binary matrices to represent subsyllable variables of specific types globally in the string. This global approach allows faster calculation of complete syllables than would be otherwise possible. Two subsyllables of the same class spanning position i to j and k to l in the string would be represented by "1"s in a binary matrix in position [i,j] and [k,l] (see fig 1). This means all matrices will be upper triangular, and most subsyllable matrices will have non-zero elements within 4 spaces from the diagonal. This allows optimization of matrix operations.

word: $_0B_1A_2T_3$
position:

"B" matrix

i \ j	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

"AT" matrix

i \ j	0	1	2	3
1	0	0	0	0
1	0	0	0	1
2	0	0	0	0
3	0	0	0	0

Fig (1)

The program begins by forming unit length matrix subsyllables by searching the input string for specific groups of phonemes. It then forms larger subsyllables from these variables such as onsets, peaks, codas, rhymes and affixes, and finally combines these to form a syllable matrix with all possible syllables in the string represented. The process of combining matrices to form larger subsyllables involves three main matrix operations:

unions of groups (binary "OR" operation), filtering groups (binary "AND" operation), and concatenation of subsyllables. This last operation is achieved by binary matrix multiplication, where multiplying a matrix with a "1" in the [i,j] position (a subsyllable spanning position i to j), with a matrix containing a [j,k] element (subsyllable from position j to k), gives a "1" in the product matrix in the [i,k] position. This corresponds to a longer subsyllable spanning the two joined subsyllables. Subsyllables not sharing a final/initial position will not produce a product element. See fig 2.

i \ j	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

"B" matrix

*

i \ j	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	0	0	0	0
3	0	0	0	0

"AT" matrix

=

i \ j	0	1	2	3
1	0	0	0	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

"BAT" matrix

Fig (2)

Further Operations

After a matrix of syllables is formed, syllables which do not form part of a connected string of syllables through the input string are removed from the matrix. Also removed are syllables which are known not to form part of the syllable dictionary (a list of which are compiled during syllable dictionary formation). The need for this second filter depends on the number of possible syllables formed from the input string. This in turn depends on the grammer rules used to form syllables. The most recent set of rules developed for the system, derived from phonemic analysis of the Macquarie dictionary, produced low numbers or syllable combinations, and made the removal of the filter routine a practical possibility. Tests were made from the new rules with filter included and removed.

A further development of the program was made upon the realization that words in a phoneme string could share phonemes as initial and final elements (e.g. "Big Game" shares the "g" phoneme). This implied that syllables had to be formed with the option of starting one phoneme earlier or ending a phoneme later, as part of syllable strings. This has the effect of removing some of the length advantage of syllable parsing, and a subsequent drop in efficiency.

RESULTS

To test the efficiency of the parser, a phonemic dictionary search was developed, and this algorithm (modified for syllable search) was attached to the syllable formation algorithm, and the syllable parse was tested against the phonemic algorithm for overall operation time, and number of file access operations. The system was implemented assuming error free input, and the results of the tests were compiled for various lengths of input string: 3,8, 12 and 17 phonemes long, made up of 10 sentences of varying complexity. Results of basic syllable parsing for unconnected word boundaries (i.e. assuming no words share a common phoneme between them) are given in table (1), and assuming phoneme sharing in table (2).

It can be seen that efficiency drops for longer input strings, this being due to the increase in syllable combinations possible for a larger matrix. A more highly developed form of this algorithm would extract string segments of 10 phonemes or less at a time and extend the matrix only when possible longer words were indicated. Systems with fast calculation abilities can expect better results from this system. Such systems would also enhance the advantage of including the syllable filter, which reduces file access time at the expense of calculation time.

The greatest block to the efficiency of this parser is the connected phoneme problem, which removes most of the advantage of syllable parsing. Further work is going on to reduce the impact of this effect, notably by evaluating the cases where phoneme sharing would not occur between syllables. If this problem can be overcome, this system may become a viable aid to CSR systems.

Table 1: No phoneme sharing

a) Using syllable filter routine

String length	3	8	12	17
Real time ratio	0.92	1.14	1.05	1.01
Search ratio	0.29	0.87	0.84	0.92

b) Without filter routine

String length	3	8	12	17
Real time ratio	0.81	1.11	1.14	1.11
Search ratio	0.29	0.93	1.05	1.04

Table 2: Phoneme sharing parsers

a) Using syllable filter routine

String length	3	8	12	17
Real time ratio	0.93	0.91	0.86	0.96
Search ratio	0.32	0.87	0.83	0.98

b) Without filter routine

String length	3	8	12	17
Real time ratio	0.72	1.19	1.2	1.24
Search ratio	0.36	0.94	1.10	1.16

Real time ratio = Syllable parser execution time/phoneme matches execution time.

Search ratio = # of syllable dictionary searches/ # of phoneme dictionary searches.

ACKNOWLEDGEMENTS

The Macquarie Library Pty Ltd, for providing a machine-readable version of the Macquarie Dictionary.

REFERENCES

[1] CHURCH, K. (1983) "Phrase Structure Parsing: A Method for Taking Advantage of Allophonic Constraints", MIT Thesis, Doctor of Philosophy.

[2] "The Macquarie Dictionary", (1981) Macquarie Library.

[3] RABINER, L.R. (1978) "Digital Processing of Speech Signals", Bell Laboratories, Inc.