AN ALGORITHM FOR CONNECTED WORD RECOGNITION

John S. Bridle, Michael D. Brown and Richard M. Chamberlain

Joint Speech Research Unit,
Princess Elizabeth Way, Cheltenham, Glos., GL52 5AJ, U.K.

ABSTRACT

The principles of an efficient one-pass dynamic programming whole-word pattern matching algorithm for the recognition of spoken sequences of connected words are described. Particular attention is given to the technique for keeping track of word-sequence decisions, which may be constrained by a finite-state syntax. Some extensions of the technique are discussed.

1. INTRODUCTION

It has been known for some time that it is possible for a machine to recognise isolated words from a small vocabulary for a known speaker, using the method of whole-word template matching [1,2]. The most successful systems use dynamic programming (DP) to cope with non-linear timescale variations [3,4,5].

Sakoe [6] has shown that it is possible to extend this method to deal with the problem of connected word recognition. His approach is to find the sequence of templates which best matches the whole input pattern allowing for timescale distortion and he presents a two-stage dynamic programming algorithm to find this best sequence. In the first stage he computes the score for every template matched against every possible portion of the input. In the second stage the best sequence of templates is computed using this information.

This paper adopts the same approach as Sakoe. Our method, described briefly in [7] and in more detail below, uses an efficient one-pass dynamic programming algorithm which, by definition, should produce the same results as Sakoe's algorithm given the same data and parameters, but uses far less computation. The recognition can be guided and improved by applying simple syntax rules. The algorithm may also be extended to operate continuously, so that the number of words in a phrase can be unlimited and the identity of the earlier words can be determined before the end of the phrase.

The JSRU algorithm is close in spirit to DRAGON [8], HARPY [9] and WBAP [10], although it is less ambitious. It can also be seen as an extension of the pioneering work of Vintsyuk [11], which has largely been overlooked in the West.

In the following 3 sections, we deal first with the relatively straightforward problem of computing the score for the best sequence of templates, and then with methods for determining the sequence itself.

2. SCORING THE BEST TEMPLATE SEQUENCE

The information about each vocabulary word is limited to one or more whole-word patterns ("templates") derived by acoustic analysis of utterances of the word by the speaker. It is assumed that the input and the template patterns are represented as sequences of units which we shall call "frames" (from vocoder terminology). Typically, each frame contains information about the short-term spectrum.

The main requirement for the matching algorithm is a measure, $d(t,i,j)$ say, of the "distance", or dissimilarity, between the i'th frame of the t'th template and the j'th input frame. The method of evaluating any particular explanation of the input (i.e. a sequence of templates, plus a time alignment between template frames and input frames) is to add up the distances between corresponding input and template frames (plus any penalties for distortions in the timescales).

The "best" sequence of templates is defined as that which can best be made to explain the input by aligning the timescales. In principle, this best sequence could be found by exhaustive search over all possible arrangements of the timescales of all possible sequences of templates. In this section we present a dynamic programming algorithm which efficiently computes the score for this best sequence of templates for a given input.

The dynamic programming score $S(t,i,j)$ is defined at any template frame and any input frame as the sum of the frame distances for the best way of aligning the first j frames of the input with any permissible sequence of templates followed by the first i frames of the t'th template (i.e. the best "time registration path" to (t,i,j)). A suitable restriction on timescale distortions, used below to illustrate the principle, is that any template frame may be repeated or single template frames may be skipped. The score for any template frame and input frame can be computed quite easily if it has already been computed for all template frames for the previous input frame:

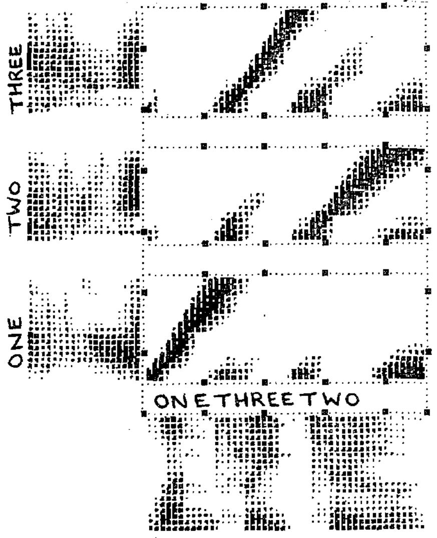
$$S(t,i,j)=\min_{a=0,1,2} \left(S(t,i-a,j-1)+d(t,i,j)+P(a) \right) \quad . (1)$$

where P(a) is a time-distortion penalty. Transitions between templates are always from the end of one template to the beginning of other templates. Thus, to compute the score corresponding to the start of each template, the ends of the preceding permissible templates must be examined and the best score selected·

$$S(t,1,j)=\min_{r\in R(t)} \left(S(r,N(r),j-1)+d(t,1,j) \right) \quad . . . (2)$$

where R(t) is the set of templates that can precede the t'th template according to the syntax and N(t) is the number of frames in the t'th template.

Computation of the scores proceeds for all templates in parallel in one forward pass through the input pattern. Although in this paper the score, S, is indexed by input frame number, in practice we need store only one value per template frame, provided that the updating in (1) and (2) is implemented carefully. The storage for the information corresponding to a template frame (e.g. the score) is called a "cell". If a template appears more than once in the syntax then each occurrence needs to be treated separately.



Connected word recognition scores using typical speech patterns

The figure above shows the DP scores computed for 3 templates and input derived from speech using a 19-channel filter bank analyser running at 50 frame/s. The whole-word templates are displayed to the left with their timescales on the vertical axis, and the input pattern is displayed at the lower edge with its timescale on the horizontal axis. The best score for each input frame is displayed as peak black, and the other scores for the same input frame are displayed relative to the best.

At the end of the input phrase, the score for the best interpretation can be found by examining the scores at the ends of all the templates that are possible at the end of a phrase. However, we are more interested in the actual sequence of templates which produced this best score. Therefore, in addition to the scores, we must also keep track of the word decisions in (2) along all the paths during the main pass over the input and then trace them back from the end of the phrase. Methods of doing this are described below.

3. PRIMITIVE SYNTAX - VINTSYUK'S ALGORITHM

Vintsyuk [11] suggests the following method of recording template sequence decisions for the simple case in which any of the R templates can follow any other. We have changed Vintsyuk's notation for consistency with the next section. The method needs three arrays which we shall call T, F and L. The best scoring template ending at the j'th input frame is defined as

$$T(j) = \operatorname{ArgMin}_{1 \leqslant r \leqslant R} \left(S(r,N(r),j) \right) \quad . . . . . (3)$$

where ArgMin means the value of the index which minimises the expression. F(j) records the input frame corresponding to the last frame of the template which precedes T(j).

The values held in the arrays F and T are sufficient to recover the best word sequence. At the end of an input phrase of length M frames, the final template in the best sequence is T(M), the last but one is T(F(M)), the next is T(F(F(M))), and so forth to the template corresponding to the beginning of the input.

In order to fill the array F, we need the third array, L. L(t,i,j) is the input frame at which the best time-registration path up to (t,i,j) ended the template previous to the t'th template. This information passes through the cells, propagating with the scores, so that

$$L(t,1,j) = j-1 \quad . . . . (4)$$

$$L(t,i,j) = L(t,i-a,j-1) \quad . . . . (5)$$

where a is the index chosen in (1).

At each input frame, the array F can now be filled with the value of L from the end of the best scoring template:

$$F(j) = L(T(j),N(T(j)),j) \quad . . . . . . (6)$$

As with the scores, it is only necessary to store the values of L for the current input frame.

4. ARBITRARY FINITE STATE SYNTAX

To deal with an arbitrary finite state syntax, we first introduce the concept of "nodes". A node is a place in the syntax where a particular set of templates is preceded by some other defined set of templates. (The "primitive syntax" of Section 3 has only one node.) We extend Vintsyuk's algorithm by applying his approach at every node. It would be possible to use two-dimensional arrays for F and T, indexed by input frame number and node number, but we use an alternative method which is particularly efficient when used with beam clipping (described in Section 6). We abandon the indexing of the one-dimensional arrays F and T by input frame, and let the arrays fill up as entries are recorded. As before, each entry in the array T records the identity of the best scoring template ending at a particular node and input frame. The array F now contains indices into array T (and into itself) which link elements of T lying on a time registration path. Thus if, at a node, an entry is made in, say, the k'th element of the array T, then the "word link" for the first frame of each template following that node satisfies

$$L(t,1,j) = k \quad \ldots\ldots\ldots\ldots\ldots (7)$$

and, as before, this word link propagates through the template according to (6). When an entry is recorded in the array T, the value of the corresponding word link, $L(t,N(t),j)$ is stored in the array F. This word link is the value of k which indexes the record of the start of the template whose end is now being recorded.

The sequence of templates for the best explanation of the input can be recovered as before by chaining down the array F and using the values $F(M)$, $F(F(M))$, $F(F(F(M)))$... as indices into the array T. These indices no longer correspond to frame numbers, so if we wish to recover the input frame numbers corresponding to the last frame of each template, another array must be added in parallel with arrays F and T to record the input frame at which each entry was made. The set of entries in the arrays F and T (and any others in parallel with them) which share the same array index is referred to below as a "word-link record".

5. EXTENSIONS TO THE ALGORITHM

One of the most difficult problems in isolated word recognition is "endpoint detection" - determining where the word starts and finishes - because this usually precedes pattern matching. Our technique avoids deciding the positions of the boundaries between words before deciding the identity of the words, and the same method is also used to deal with the phrase endpoint problem. We simply use phrase syntaxes which start and end with "silence" templates in loops so that they can match silence of any length.

The algorithm described above is suitable for recognising isolated phrases of limited length, but it can be extended to deal with continuous speech. This removes the need to even detect the presence of a phrase before pattern matching, and it means that some words may be recognised before the speaker has finished the phrase. Periodically, the word-link records that are referenced by "active cells" (i.e. cells currently being processed) are traced back through the array F to the word link record at which they all meet. The words before this word link record are unambiguous in that no further input can change our decision about these words. The recogniser can operate continuously by outputting this unambiguous path and then freeing the storage in the word-link record list, by removing the unambiguous path records and the word-link records that cannot be accessed from the active cells. Spohrer et. al. [12] describe a related continuous operation algorithm.

In isolated word recognisers, there is usually provision for rejecting an utterance for which all the templates score worse than some preset threshold. For connected word recognition the equivalent is to reject a portion of an utterance while accepting the rest. Our method is to have an elastic "pseudo-template" which always produces the same "distance" when matched with any input frame. By incorporating this "wildcard" template into the syntax, spurious inputs, such as breath noises and illegal words, can be permitted at selected points in the syntax. The distance for the wildcard template has to be set carefully to make it unlikely that the wildcard template will be chosen when a permitted vocabulary word is spoken.

6. DISCUSSION

The above algorithm will by definition find the best "explanation" of the input pattern in terms of a sequence of template patterns but it may seem rather expensive in computation and storage. However, compared with an isolated word recogniser with the same size vocabulary, using the same methods of pattern representation and dynamic programming matching algorithm, it takes about the same amount of computation per input frame. Isolated word recognisers usually wait until the end of the input, then match each template in turn. However, we consider all templates in parallel, and therefore the amount of working storage for the scores is larger by a factor of the vocabulary size. The word links, in the array L, and the arrays F and T also need extra storage, but the total amount of working storage will normally be much less than that required to hold the template patterns themselves.

The amount of computation can be significantly reduced by "pruning" the DP scores

(Lowerre's "Beam Search" [9]). For each input frame, all scores which are more than some "beam width factor" away from the best score for that input frame are removed from further consideration. This avoids considering relatively unlikely interpretations of the input pattern, but keeps the options open where there is some ambiguity. The range of numbers needed to represent the scores can also be reduced by setting the best score for each input frame to zero, by subtraction.

A possible disadvantage of our algorithm, compared with the two-stage algorithm of Sakoe [6] and the level-building algorithm of Myers and Rabiner [14], is that there is no direct way to constrain the recognition process so that only solutions with a specified number of words (e.g. 4 digits) are found. However, it is not difficult to constrain our system to recognise a fixed number of words by using a suitable syntax. Myers and Levinson [15] show how the level-building algorithm can be extended to deal with finite state syntax without loops, but the type of finite state syntax (with loops) that we use is more general than appears possible for the above algorithms.

7. CONCLUSIONS

The above technique should be useful in many applications of simple connected word voice input to machines.

We have so far done only rather limited experiments using small vocabularies. However our results have shown that our algorithm produces results similar to those produced by machines based on Sakoe's algorithm on the same data.

The full power and the limitations of these methods have yet to be ascertained. A real-time hardware equipment based on this algorithm has recently been designed and constructed under contract by Logica Ltd., of London [13]. This equipment has storage for more than 100 templates, accepts syntax definitions and operates continuously. It also incorporates several refinements in acoustic analysis and frame distance computation and retains auxiliary information for access by a host computer. This equipment, and further models, will be used to explore both the template matching technique itself and the applications of speech recognition in many military and civil areas.

8. REFERENCES

[1] P.Denes and M.V.Matthews, "Spoken digit recognition using time-frequency pattern matching", J. Acoust. Soc. Am. 32(11), pp.1450-1455, 1960.

[2] L.C.W.Pols, "Real-time recognition of spoken words", IEEE Trans. Comput., Vol.C20, pp.972-978, September 1971.

[3] T.K.Vintsyuk, "Speech recognition by dynamic programming methods", Kibernetika (Cybernetics), No.1, 1968.

[4] Z.M.Velichko and N.G.Zagoruyko, "Automatic recognition of 200 words", Int. J. Man-Machine Studies, 2(3), pp.222-234, 1970.

[5] F.Itakura, "Minimum prediction residual principle applied to speech recognition", IEEE Trans. Acoustics, Speech and Signal Processing, Vol.ASSP-23, pp.67-72, February 1975.

[6] H.Sakoe, "Two-level DP matching - A dynamic programming based pattern matching algorithm for connected word recognition", IEEE Trans. Acoustics, Speech and Signal Processing, Vol.ASSP-27, No.6, pp.588-595, December 1979.

[7] J.S.Bridle and M.D.Brown, "Connected word recognition using whole word templates", Proc. Institute of Acoustics, Autumn Conference, pp.25-28, November 1979.

[8] J.K.Baker, "The DRAGON system - an overview", IEEE Trans. Acoustics, Speech and Signal Processing, Vol.ASSP-23, No.1, pp.24-29, February 1975.

[9] B.T.Lowerre, "The HARPY speech recognition system", Carnegie-Mellon University, Dept. Computer Science (Dissertation), April 1976.

[10] R.Bakis, in F.Jelinek, "Continuous speech recognition by statistical methods", Proc. IEEE, Vol.64, No.4, Sections IX and X, pp.548-552, April 1976.

[11] T.K.Vintsyuk, "Element-wise recognition of continuous speech consisting of words of a given vocabulary", Kibernetika (Cybernetics), No.2, 1971.

[12] J.C.Spohrer, P.F.Brown, P.H.Hochschild and J.K.Baker, "Partial traceback in continuous speech recognition", Proc. Int. Conf. Cybernetics and Society, Boston, MA, USA, October 1980.

[13] J.B.Peckham, J.R.D.Green, J.V.Canning and P.Stephens, "A real-time hardware continuous speech recognition system", 1982 Int. Conf. Acoustics, Speech and Signal Processing, May 1982.

[14] C.S.Myers and L.R.Rabiner, "A level building dynamic time warping algorithm for connected word recognition", IEEE Trans. Acoustics, Speech and Signal Processing, Vol.ASSP-29, pp.284-297, April 1981.

[15] C.S.Myers and S.E.Levinson, "Connected word recognition using a syntax-directed dynamic programming temporal alignment procedure", Proc. Int. Conf. Acoustics, Speech and Signal Processing, Vol.3, pp.956-959, April 1981.