ARCHIVES OF ACOUSTICS
16, 3–4, 475–487 (1991)

ALGORITHMS FOR CONNECTED WORD RECOGNITION – A GLOBAL APPROACH

S. GROCHOLEWSKI

Institute of Computing Science Technical University of Poznań
(60-965 Poznań, ul. Piotrowo 3a)

In this paper, which is basically a review, three approaches to the problem of recognizing word connected speech are presented. They all use the dynamic programming technique to solve the optimization problem which occurs in matching an unknown utterance against an artificially synthesized sequence of word templates. They are compared with regard to the computation time and the memory requirement.

1. Introduction

In connected word recognition the spoken input is a sequence of words from a limited vocabulary, often uttered without pauses in-between, and the recognition is based on matching isolated word reference templates. The general idea of a global approach consists in comparing artificially synthesized connected reference patterns with an unknown utterance. For a 100-word vocabulary and the maximum of 5 words in an utterance, this approach involves 10^{10} comparisons. Such an unacceptably great number of calculations can be efficiently reduced by the dynamic programming (DP) technique successfully used in isolated word recognition systems [4], [11].

In this paper, three algorithms using the DP technique are presented, concerning three approaches characteristic of the first significant works undertaken in Japan (algorithm I), the USA (algorithm II) and the USSR (algorithm III). The comparisons will be made in terms of both computation and storage requirements.

Let us introduce the basic terms used in the description of these algorithms. Let X signify a sequence of I vectors describing an unknown utterance to be recognized. Let the vocabulary comprise N templates: $W(1), \ldots, W(N)$, and let the n-th template be described by the sequence of $J(n)$ elements vectors.

Let:

– $d(i, j, n)$ signify the local distance between the i-th element of an utterance and the j-th element of the n-th template;

– $D(i, j, n)$ signify the accumulated distance to the point determined by the following coordinates: the i-th element of an utterance and the j-th element of the n-th template from the beginning of the uterance.

2. Algorithm I (Two-Level DP-Matching Algorithm [10])

Let $C(l, m)$ be a partial pattern as the part of the utterance X which begins with the l-th element and ends with the m-th one. At the first level of the algorithm (word level), for all the combinations (l, m) such that $1 \leqslant l < m \leqslant I$ considering all the templates $W(n)$, where $n = 1 \div N$, it is necessary to find:

$$D^*(l, m) = \min_n D\big(C(l, m), W(n)\big) = \min_n D(l, m, n) \tag{1}$$

$$N^*(l, m) = \operatorname*{argmin}_n D\big(C(l, m), W(n)\big) = \operatorname*{argmin}_n D(l, m, n) \tag{2}$$

where $D^*(l, m)$ – distance between the partial pattern $C(l, m)$ of the utterance X and the nearest template, $N^*(l, m)$ – the nearest template for the $C(l, m)$.

The operator "$\operatorname*{argmin}_n D(\ )$" means to find the optimum argument n which minimizes $D(\ )$.

In practice, for every l, conceived of as the beginning of some word, and for the length of its template equal to $J(n)$, the range of m can be determined as in the inequality (3) below:

$$l + J(n) - r \leqslant m \leqslant l + J(n) + r. \tag{3}$$

This is possible assuming that the length of the word differs from the length of its template by the value of r.

The range of m can be determined as demonstrated in Fig. 1a. For every l, with respect to the inequality (3), only $2r+1$ pairs $\{D^*(i, m), N^*(l, m)\}$ should be designated. The conditions expressed in the inequality (3) also cause the limitations of maximum l: $l_{\max} = I - J(n) + r$ (cf. Fig. 1b).

Figure 2 presents the set of all possible pairs (l, m) for a simple example, where $I = 9, r = 1, J(n) = 3$ for all n. These pairs are represented by the lines connecting the respective points l, m.

After all $N^*(l, m)$ and $D^*(l, m)$ for every pair (l, m) have been found, it is necessary to find, at the second level, such a combination K^* of pairs $\{(l_1 m_1), \ldots, (l_k, m_k), \ldots, (l_{K^*}, m_{K^*})\}$, where

$$l_1 = 1, \quad l_k = m_{k-1} + 1, \quad m_{K^*} = I, \tag{4}$$

for which the sum of distances $D^*(l_k, m_k)$, $k = 1 \div K^*$ henceforth designated by $T_{K^*}(I)$, will be minimum.

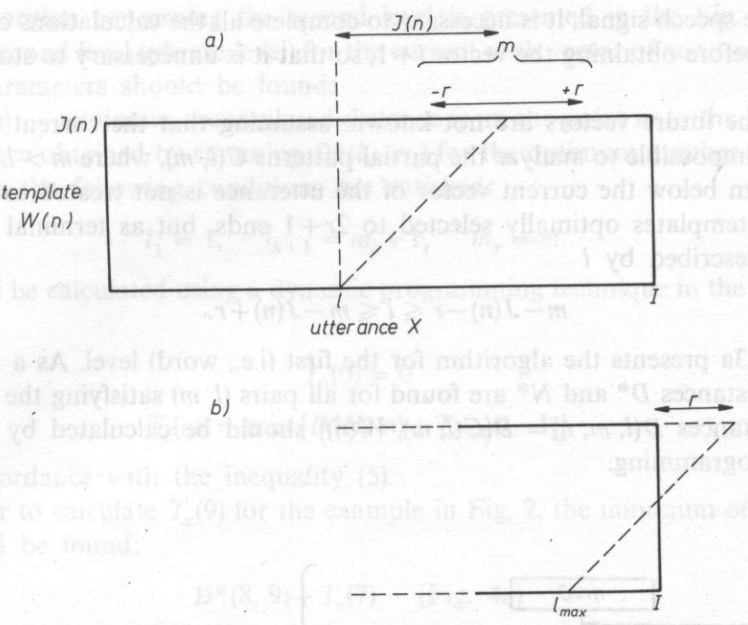

FIG. 1. Illustration of the way in which the range of m can be determined

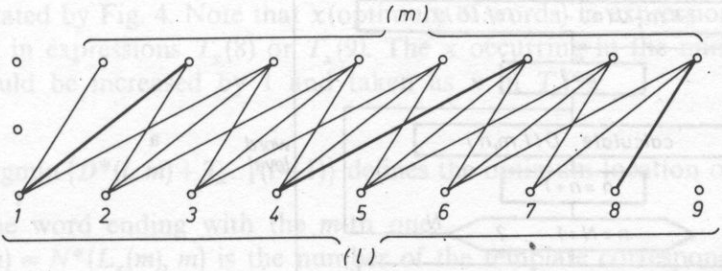

FIG. 2. The set of all possible pairs (l, m) for a simple example, where $I = 9$, $r = 1$, $J(n) = 3$

In Fig. 2 the thick lines present an example of a combination of the 3 pairs $\{(1, 3), (4, 7), (8, 9)\}$ satisfying the conditions (4). Let $K\max$ be the maximum expected number of words in utterance. The optimum number of words is defined as

$$K^* = \underset{K}{\operatorname{argmin}} \{T_K(I)\}, \qquad \text{where } K = 1 \div K_{\max}.$$

On the basis of both the optimum combination K^* of the pairs (l_k, m_k), where $k = 1 \div K^*$, and (2), the sequence of words in this utterance can be easily determined.

Now let us describe such an algorithm which can be useful in a speech recognition system working in real time. This means that after obtaining the i-th

vector of the speech signal, it is necessary to complete all the calculations concerning this vector before obtaining the vector $i+1$, so that it is unnecessary to store the i-th one.

Since the future vectors are not known, assuming that the current is the l-th vector, it is impossible to analyze the partial patterns $C(l, m)$, where $m > l$. Hence, in the algorithm below the current vector of the utterance is not treated as initial for a group of templates optimally selected to $2r+1$ ends, but as terminal for $2r+1$ beginning described by l

$$m-J(n)-r \leqslant l \leqslant m-J(n)+r. \tag{5}$$

Figure 3a presents the algorithm for the first (i.e., word) level. As a result, the minimum distances D^* and N^* are found for all pairs (l, m) satisfying the condition (5). The distances $D(l, m, n) = D(C(l, m), W(n))$ should be calculated by means of dynamic programming.

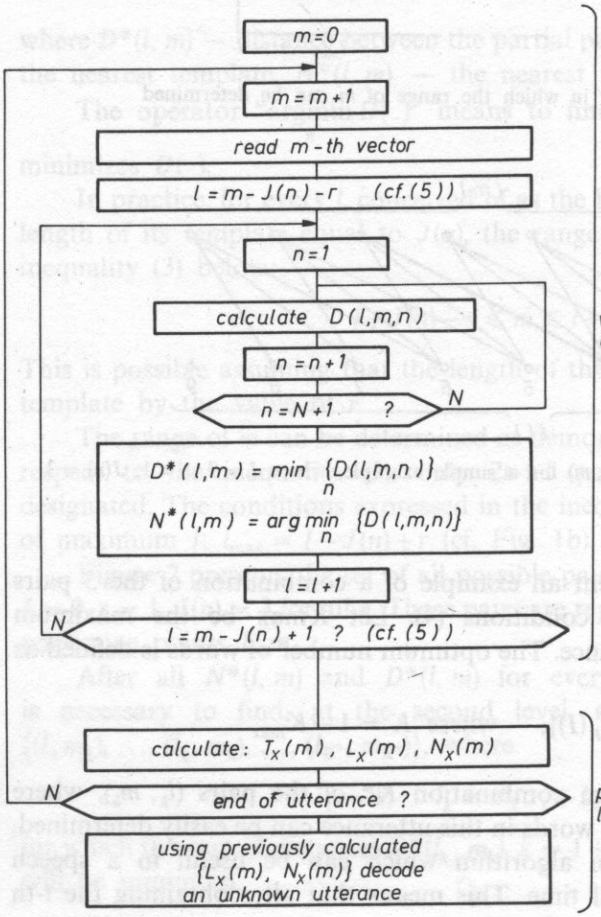

FIG. 3. Description of algorithm *I* for a real time recognition system

The algorithm concerning the second level is presented in the Fig. 3b.

At the second level (phrase level) for the current m-th vector of an utterance, the following parameters should be found:

– $T_x(m)$ – minimum accumulated distance from the point m to the beginning of an utterance obtained by summing $D^*(l_k, m_k)$ for the optimum number of x pairs (l_k, m_k) when the following conditions are satisfied:

$$l_1 = 1, \quad l_{k+1} = m_k + 1, \quad m_x = m. \tag{6}$$

$T_x(m)$ should be calculated using a dynamic programming technique in the following way:

$$T_0(0) = 0$$
$$T_x(m) = \min_l \{D^*(l, m) + T_{x-1}(l-1)\} \tag{7}$$

for l in accordance with the inequality (5).

In order to calculate $T_x(9)$ for the example in Fig. 2, the minimum of the three sums should be found:

$$D^*(8, 9) + T_x(7) \quad \text{(Fig. 4a)} \tag{8}$$

$$D^*(7, 9) + T_x(6) \quad \text{(Fig. 4b)} \tag{9}$$

$$D^*(6, 9) + T_x(5) \quad \text{(Fig. 4c)} \tag{10}$$

This is illustrated by Fig. 4. Note that x(optimum of words) in expression $T_x(7)$ can differ from x in expressions $T_x(8)$ or $T_x(9)$. The x occurring in the minimum sum (8)–(10) should be increased by 1 and taken as x in $T_x(9)$.

(11)

$-L_x(m) = \underset{l}{\operatorname{argmin}} \{D^*(l, m) + T_{x-1}(l-1)\}$ defines the optimum location of the initial vector for the word ending with the m-th one.

– $N_x(m) = N^*(L_x(m), m)$ is the number of the template corresponding to the partial pattern $C(L_x(m), m)$ calculated according to Eq. (2) at the first level.

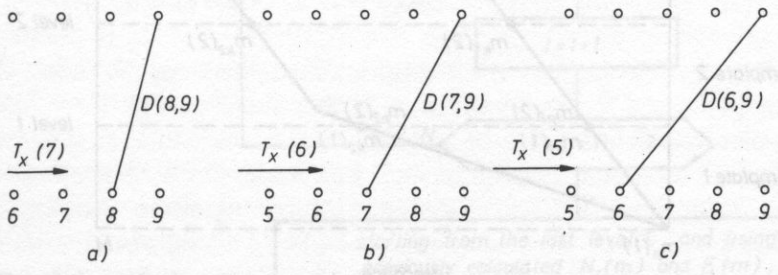

FIG. 4. Three ways of calculating $T_x(9)$ for the example in Fig. 2

The decision about the recognition of the whole utterance is generated on the basis of the pairs $\{N_x(m), L_x(m)\}$ for $m = 1 \div I$ in the manner shown in Fig. 5. The last column in Fig. 5 indicates that the last word is the word $N_{K^*}(I)$, the penultimate one ends for $m = L_x(I) - 1$ and that this is the word $N_x(L_x(I) - 1)$, etc.

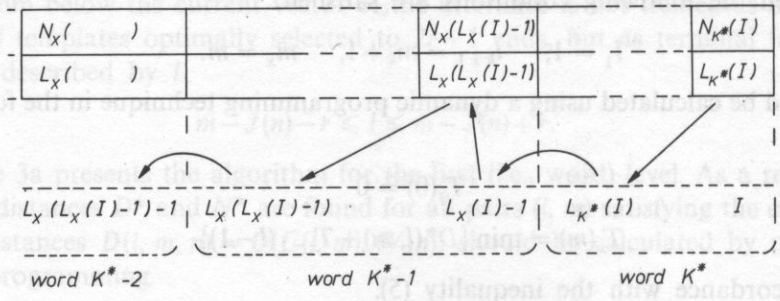

FIG. 5. The manner of decoding of an unknown utterance

3. Algorithm II (Level Building DTW Algorithm [6, 2, 8]

Let us assume that the utterance is a sequence of L words. On the basis of L and the duration (I) of the utterance, it is possible to determine the range of the expected location of the beginning $(m_{p1}(l), m_{p2}(l))$ and the location of the end $(m_{k1}(l), m_{k2}(l))$ of the l-th word. The manner in which they can be determined has been illustrated in Fig. 6 for the case when $L = 4$. The horizontal lines separating the templates define the so-called levels, henceforth they are designated by the letter l; the l-th level concerns

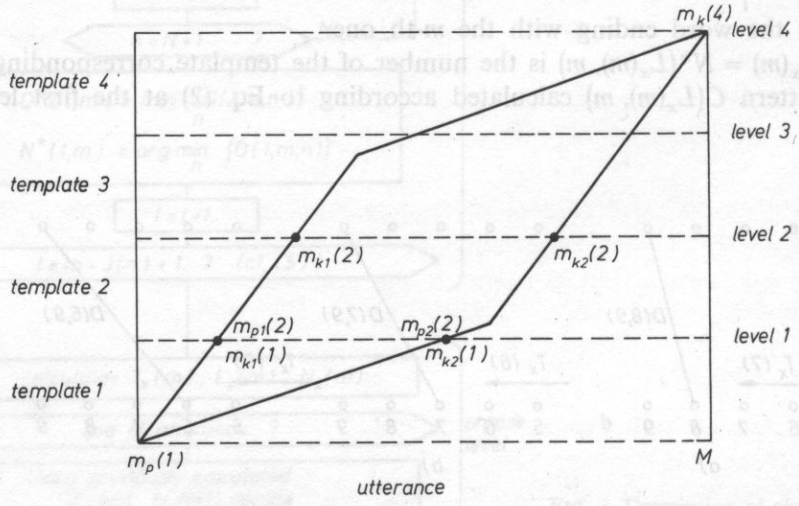

FIG. 6. The presumed beginnings of the successive words in an unkown utterance

the end of the l-th template. The solid lines limit the area for the warping function. The intersections of the lines indicate the possible location of both the end of the previous word and the possible location of the beginning of the subsequent one.

Figure 7 presents the procedure for the recognition of an utterance consisting of L words. This procedure is exemplified in Fig. 8. It has been assumed that the utterance is composed of L words belonging to the following vocabulary $\{A, B, C, D, E, F\}$.

At each level, a pair of $N_l(m)$ for for m satisfying

$$m_{k1}(l) \leqslant m \leqslant m_{k2}(l) \tag{13}$$

has been found. The description of, for instance, a point at the level $l = 3$ of the type $F/4$ means that the optimum warping function passes through the previous level at the point $m = 4$, and that the part of the utterance corresponding to the 3-rd word is the least distant from the template F.

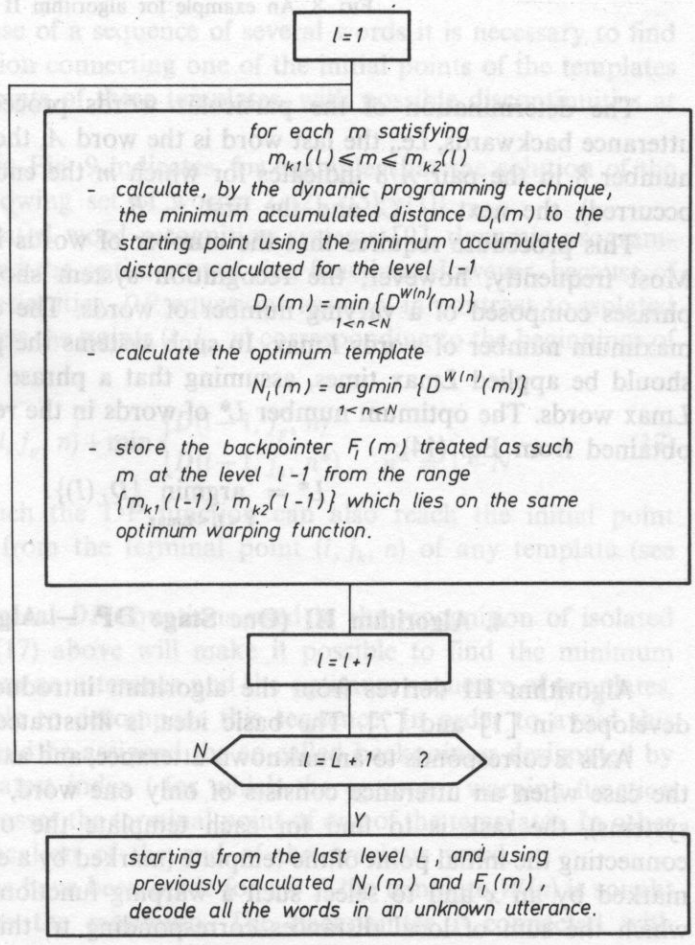

FIG. 7. Description of algorithm II

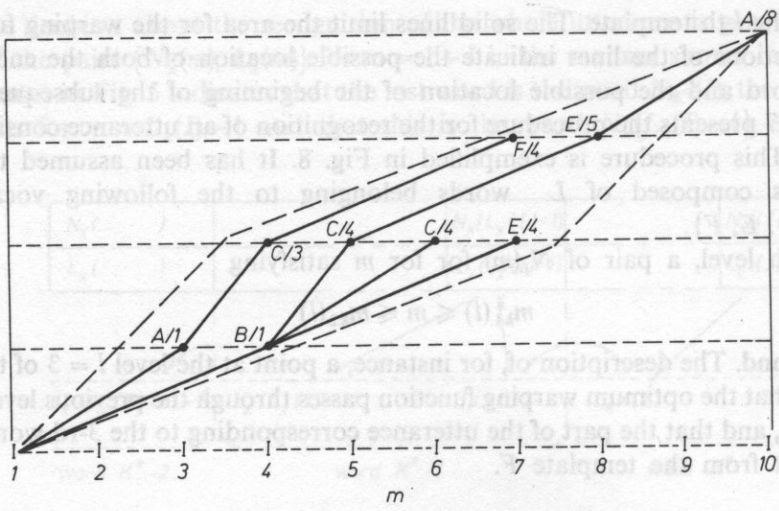

FIG. 8. An example for algorithm II

The determination of the particular words proceeds from the end of the utterance backwards, i.e., the last word is the word A, the penultimate one is E (the number 8 in the pair $A/8$ indicates for which m the end of the penultimate word occurred), the next $-$ C, and the first $-$ B.

This procedure requires that the number of words in an utterance be known. Most frequently, however, the recognition system should be able to recognize phrases composed of a varying number of words. The only limitation can be the maximum number of words $L\max$. In such systems the procedure described above should be applied $L\max$ times, assuming that a phrase may consist of 1, 2, ... to $L\max$ words. The optimum number L^* of words in the recognized utterance can be obtained from Eq. (14):

$$L^* = \underset{1<L<L\max}{\operatorname{argmin}} \{D_L(I)\}. \tag{14}$$

4. Algorithm III (One Stage DP $-$ Algorithm [7])

Algorithm III derives from the algorithm introduced by VINTSUK [12] and developed in [1] and [7]. The basic idea is illustrated in Fig. 9.

Axis x corresponds to an unknown utterance, and axis y to a set of templates. In the case when an utterance consists of only one word, (isolated word recognition systems), the task is to find for each template the optimum warping function connecting the initial point of the template (marked by a circle) with is terminal point marked by an x and to select such a warping function (i.e., such a template) for which the sum of local distances corresponding to this function is minimum.

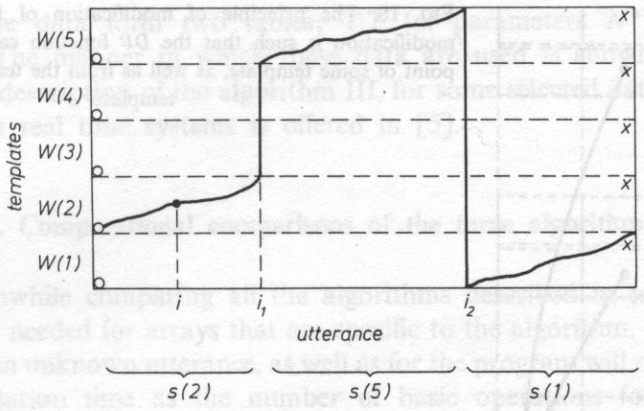

FIG. 9. An example of the optimal warping function for the sequence $\{s(2), s(5), s(1)\}$; $s(n)$ signifies the part of an unknown utterance corresponding to the n-th template

In the more difficult case of a sequence of several words it is necessary to find the optimum warping function connecting one of the initial points of the templates with one of the terminal points of these templates, with possible discontinuities at their boundaries.

The warping function in Fig. 9 indicates, for example, that the solution of the recognition task is the following set of words: $\{s(2), s(5), s(1)\}$.

Similarly as in the isolated word recognition systems [9], dynamic programming can also be used to find the optimum warping function. However, because of the above-mentioned discontinuities, *DP* equations assume, in contrast to isolated words [9], a modified form for the points (i, j_p, n) corresponding to the beginnings of the templates:

$$D(i, j_p, n) = d(i, j_p, n) + \min \begin{cases} D(i-1, j_p, n) \\ D(i-1, j_k, n^*) \end{cases} \quad n^* = 1 \div N \tag{17}$$

The modification is such the DP function can also reach the initial point (i, j_p, n) of some template, from the terminal point (i, j_k, n) of any template (see Figs. 9 and 10).

The application of classical *DP* equations used in the recognition of isolated words modified as in Eq. (17) above will make it possible to find the minimum accumulated distance between an utterance and the optimum sequence of templates, but will not make it possible to decompose this sequence. In order to avoid this difficulty, each $D(i, j, n)$ should be assigned the so-called backpointer designated by i'. It is defined as such greatest index i for which the optimum warping function reaching the point (i, j, n) crosses the terminal point of one of the templates. In other words, it determines the location of the end of the previous word.

After all the calculations have been made for $i = I$, the template $W(n)$ is sought for which $D(I, j_k, n)$ attains the minimum. The backpointer i' connected with

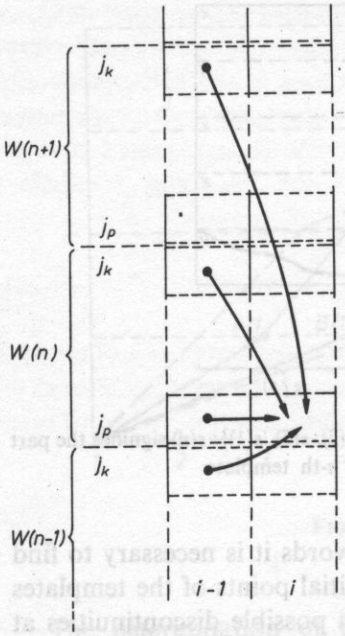

FIG. 10. The principle of modification of *DP* equations: the modification is such that the *DP* function can reach the initial point of some template, as well as from the terminal point of any template

$D(I, j_k, n)$ indicates the end of the previous word. The parameter n for which $D(i', j_k, n) = \min$ identifies this word. By repeating this procedure until $i = 1$, we can find the optimum sequence of words.

From the practical point of view, in order to simplify the above procedure, it is recommended to store for each i the following two data: parameter n, for which $D(i, j_k, n) = \min$, i.e., N^*, and the backpointer i^* connected with the minimum

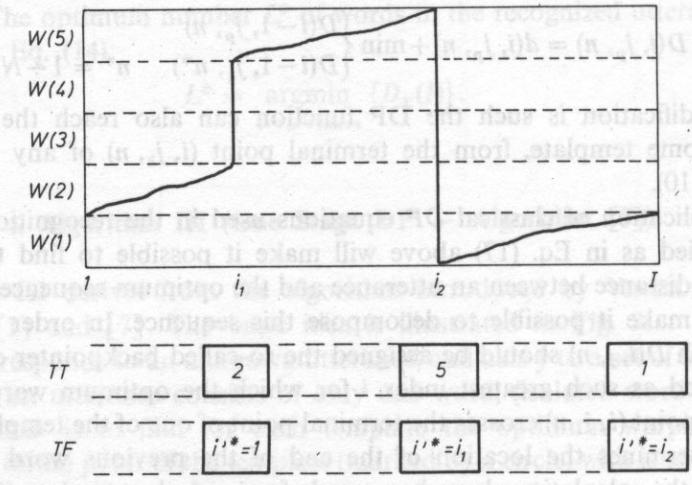

FIG. 11. The manner in which the data in the tables *TT* and *TF* (algorithm III) are used

$D(i, j_k, n)$. These data form two tables: TT (of parameters N^*) and TF (of backpointers). The manner in which these data are used is shown in Fig. 11.

A detailed description of the algorithm III, for some selected data, which could be employed in real time systems is offered in [5].

5. Computational comparisons of the three algorithms

It is worthwhile comparing all the algorithms described in terms of:

– storage needed for arrays that are specific to the algorithm. Storage for the templates and an unknown utterance, as well as for the program will not be included,

– computation time as the number of basic operations for the dynamic programming technique, i.e., calculations of $D(i, j)$ together with $d(i, j)$.

Algorithm I:

– storage

At the first level for each m it is necessary to calculate and store $(2r+1)$ values of $D^*(l, m)$ and $N^*(l, m)$. Thus the first level requires $2I(2r+1)$ memory locations. For calculating $(2r+1)$ pairs of $\{D^*(l, m), N^*(l, m)\}$ for a given m, only a very small number $2r+1$ of memory locations is sufficient. Nevertheless for real time systems, when the calculations at the two levels are performed simultaneously, and after the $(2r+1)$ pairs of $(D^*(m), N^*(m))$ have been determined for a given m, it is necessary to calculate $T_x(m)$, x, $N_x(m)$ additionally. From amongst these data only $(2r+1)$ pairs of $\{T_x(m), x\}$ and all I pairs of $\{L_x(m), N_x(m)\}$ should be stored.

The above considerations indicate that the theoretically minimum number of memory locations for a real time system amounts to $3(2r+1)+2I$. It is worth mentioning, what is also important for the two other algorithms, that the application of only a minimum number of memory locations can decrease the speed of the program.

– computation time

At the first level, for each l, where $l = 1 \div I$, and for each template $W(n)$, where $n = 1 \div N$, the calculation of $D(i, j)$ must be repeated $(2r+1)J_{av}$ times, where J_{av} is an average template length. Hence the number of these calculations equals $INJ_{av}(2r+1)$.

At the phrase level, I dynamic programming equations should be solved, which makes the total number of calculations of $D(i, j)$ at both levels equal to

$$I(NJ_{av}(2r+1)+1) \approx INJ_{av}(2r+1).$$

Algorithm II:

– storage

It theoretically requires minimum $J\max$ memory locations for $D_l(m)$ calculations and $3IL\max$ locations for storing $D_l(m)$, $N_l(m)$, $F_l(m)$

– computation time

In [6] and [7] various algorithms are compared by means of a number of local

distance ($d(i, j)$) calculations. The computations time for algorithm II is given in the quoted works as the number of $d(i, j)$ calculations, i.e., $N L J_{av} I/3$.

Algorithm III

– storage

Algorithm III requires:

– $2NJ_{av}$ memory locations for given i for $D(i, j, n)$ and $i'(i, j, n)$,

– $2I$ memory locations to store N^* and i'^* for every i; hence the total requirement is $2I + J_{av}N$,

– computation time.

The number of DP equations is $NJ_{av}I$.

In order to compare all the described algorithms in terms of storage and computation time, let us consider an example of a recognition system with the following parameters:

– the number of templates $N = 64$,

– the maximum length of an unknown utterance $I = 256$ as the number of vectors describing the speech signal; for a 20 ms frame, this gives ca 5 sec. of speech,

– $r = 8$,

– the average number of vectors in a template equals 32,

– the maximum number of words in a phrase $L_{\max} = 5$.

The storage and computation time requirements for the above example are shown in Fig. 12. It should be emphasized at this point that the quantity of the memory locations required is so small that it does not pose a problem for modern digital technology. It therefore seems that the critical parameter in real time recognition systems will be the computation time. With regard to this, algorithm III seems to be most suitable.

	algorithm I	algorithm II	algorithm III
storage	566	3872	4608
computation time	8.912.896	873.813	524.286

FIG. 12. The comparison of the three algorithms for an example given in the text

8. Conclusions

In this paper three approaches to the problem of connected word recognition are presented. They all use the dynamic programming technique to solve the optimization problem which occurs while comparing an unknown utterance with an artificially synthesized sequence of word templates.

Although algorithm III seems to be most appropriate with regard to the most critical parameter, i.e., computation time, and therefore was implemented in special

purpose chips [3], it is worthwhile mentioning that algorithm I has been used in the first commercial connected word recognition system DP-100 [11].

An experimental system based on algorithm III, worked out and tested by the author, will be presented in [5].

References

[1] J. Bridle, M. Brown, R. Chamberlain, *An Algorithm for Connected Word Recognition*, Proc. ICASSP 1982, pp. 899–902.

[2] Ch. Gagnoulet, M. Couvrat, *SERAPHINE – A Connected Words Speech Recognition System*, Proc. ICASSP 1982, pp. 887–890.

[3] S. Glinski et al., *The Graph Search Machine (GSM): A VLSI Architecture for Connected Speech Recognition and Other Applications*, Proc. of the IEE, 75, 9, pp. 1172–1184 (1987).

[4] S. Grocholewski, *Dynamic Programming in Automatic Speech Recognition*, (in Polish) Elektrotechnika (AGH), 4, 1, pp. 106–126 (1985).

[5] S. Grocholewski, *The Method for Connected Word Recognition*, in preparation for Archives of Acoustics.

[6] C. S. Myers, L. R. Rabiner, *A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition*, IEEE Trans. ASSP, 29, 2, pp. 284–297 (1987).

[7] H. Ney, *The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition*, IEEE Trans. ASSP, 32, 2, pp. 263–271 (1984).

[8] L. R. Rabiner, J. G. Wilpon, F. K. Soong, *High Performance Connected Digit Recognition Using Hidden Markov Models*, Proc. ICASSP 1988, pp. 119–122.

[9] H. Sakoe, S. Chiba, *Dynamic Programming Algorithm Optimization for Spoken Word Recognition*, IEEE Trans. ASSP, 26, 2, pp. 43–49 (1978).

[10] H. Sakoe, *Two-Level DP-Matching – A Dynamic Programming Based Pattern Matching Algorithm for Connected Word Recognition*, IEEE Trans. ASSP, 27, 6, pp. 588–595 (1979).

[11] S. Tsuruta, H. Sakoe, S. Chiba, *DP-100 Connected Speech Recognition System*, presented at INTELCOM 1979.

[12] Т. К. Винцук, *Поэлементное распознавание непрерывной речи составленной из слов заданного словаря*, Кибернетика, 7, 2, pp. 133–143 (1971).

Received January 10, 1990