AT&T BELL LABORATORIES TECHNICAL JOURNAL
Vol. 63, No. 3, March 1984
Printed in U.S.A.

On the Application of Embedded Training to Connected Letter Recognition for Directory Listing Retrieval

By L. R. RABINER,* J. G. WILPON,* and S. G. TERRACE*

(Manuscript received August 26, 1983)

Automatic speech recognition has advanced to the stage where it is now possible to recognize connected strings of words (e.g., digits, letters, city names, airline terms) from a word reference set of isolated tokens of each of the words in the vocabulary. Recently, an improved training technique called embedded word training was proposed, in which reference word patterns were extracted from within connected word sequences themselves. In this investigation we extend the embedded word training procedure to handle letters of the alphabet for use in a directory listing retrieval task. By performing connected letter recognition of spoken names based on letter classes (rather than specific letters themselves), we show how reliable name recognition results can be achieved using a fairly straightforward system on 200 randomly chosen names (chosen from an 18,210-name directory) spoken at a normal rate by four talkers (three male, one female) in a speaker-trained mode. We have found that an 8-percent improvement in name recognition accuracy is obtained when using embedded letter training patterns over that obtained from isolated letter patterns alone. The overall name recognition accuracy was close to 95 percent.

I. INTRODUCTION

Research in the area of isolated word recognition has progressed to the state where a wide variety of practical recognition systems exist both in the laboratory and in the commercial world.[1–7] These systems

* AT&T Bell Laboratories.

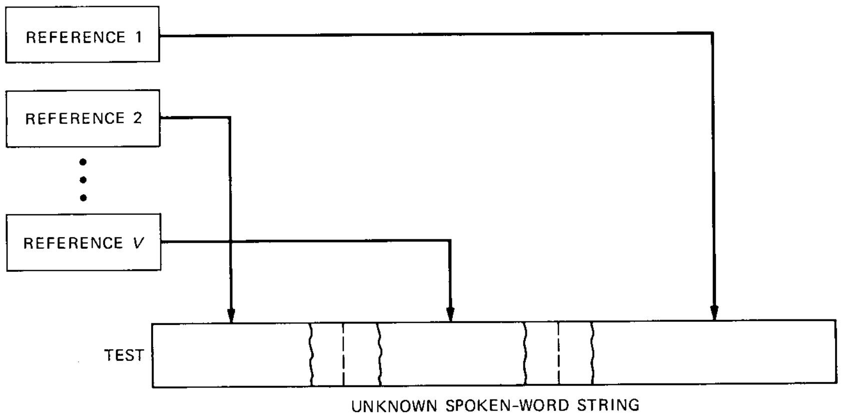

Fig. 1—Illustration of connected word recognition by concatenation of individual reference patterns.

are often capable of handling medium- to large-size (100- to 1000-word) vocabularies; they can work in both speaker-trained and speaker-independent modes; they can work over dialed-up (local) telephone lines; and they can take advantage of task syntax to improve overall system accuracy. The major shortcoming of these recognition systems is the isolated word format itself, since it is highly unnatural for use in a wide variety of tasks (e.g., digit dialing, word spelling, etc.).

The area of connected word recognition has made great strides forward in the last few years, and it has reached the point where there are several laboratory and commercial systems that attain some limited degrees of success.[8-11] Figure 1 summarizes the basic idea in a pattern-based approach to connected word recognition. Assume we are given a test pattern, **T**, which represents an unknown spoken-word string, and we are given a set of V reference patterns, $\{R_1, R_2, \cdots, R_v\}$, each representing some word of the vocabulary. The connected word recognition problem consists of finding the "super" reference pattern $\mathbf{R}^s$,

$$\mathbf{R}^s = R_{q(1)} \oplus R_{q(2)} \oplus \cdots \oplus R_{q(L)}. \tag{1}$$

This is the concatenation of L reference patterns, $R_{q(1)}$, $R_{q(2)}, \cdots, R_{q(L)}$, which best matches the test string, **T**, in the sense that the overall distance between **T** and $\mathbf{R}^s$ is minimum over all possible choices of L, $q(1)$, $q(2)$, $\cdots$, $q(L)$, where the distance is an appropriately chosen distance measure.

There are several problems associated with solving the above connected word recognition problem. First, we don't know L, the number of words in the word string. Hence, our proposed solution must provide the best matches for all reasonable values of L, e.g., $L = 1, 2, \cdots, L_{\mathrm{MAX}}$. Second, we don't know, nor can we reliably find, word boundaries, even when we have postulated L, the number of words in the string. The implication of this observation is that our word recognition algorithm must work without direct knowledge of word boundaries; in fact, the estimated word boundaries will be shown to be a byproduct of the matching procedure. The third problem with a template matching approach is that the word matches are generally much poorer at the boundaries than at frames within the word. In general, this is a weakness of word-matching schemes, which can be somewhat alleviated by the matching procedures that can apply lesser weight to the match at template boundaries than at frames within the word. A fourth problem is that word durations in the string are often grossly different (shorter) from the durations of the corresponding reference patterns. To alleviate this problem, one can use some time prenormalization procedure[12] to warp the word durations accordingly, or rely on reference patterns extracted from embedded word strings, as will be described later in this paper. Finally, the last problem associated with matching word strings is that the combinatorics of matching strings exhaustively (i.e., by trying all combinations of reference patterns in a sequential manner) is prohibitive.

A number of different ways of solving the connected word recognition problem, which avoid the plague of combinatorics mentioned above, have been proposed. Among these algorithms are the two-level Dynamic Programming (DP) approach of Sakoe,[8] the level-building approach of Myers and Rabiner,[9] the parallel single-stage approach of Bridle et al.,[10] and the nonuniform sampling approach of Gauvain and Mariani.[11] Although each of these approaches differs greatly in implementation, all of them are similar in that the basic procedure for finding $\mathbf{R}^s$ is to solve a time-alignment problem between $\mathbf{T}$ and $\mathbf{R}^s$ using Dynamic Time Warping (DTW) methods.

Figure 2 illustrates the level-building DTW-based approach to connected word recognition. Shown in this figure are the warping paths for all possible length matches to the test pattern, along with the implicit word boundary markers $(e_1, e_2, \cdots, e_{L-1}, e_L)$ for the dynamic path of the L-word match. The level-building algorithm builds up all possible L-word matches one level (word in the string) at a time. For each string match found, a segmentation of the test string into appropriate matching regions for each reference word in $\mathbf{R}^s$ is obtained. In addition, for every string length L, the best β matches (i.e., the β lowest-distance L-word strings) can be found. The details of the actual level-building algorithm are available elsewhere,[9] and will not be discussed here. Instead, we will rely on the properties of the algorithm,

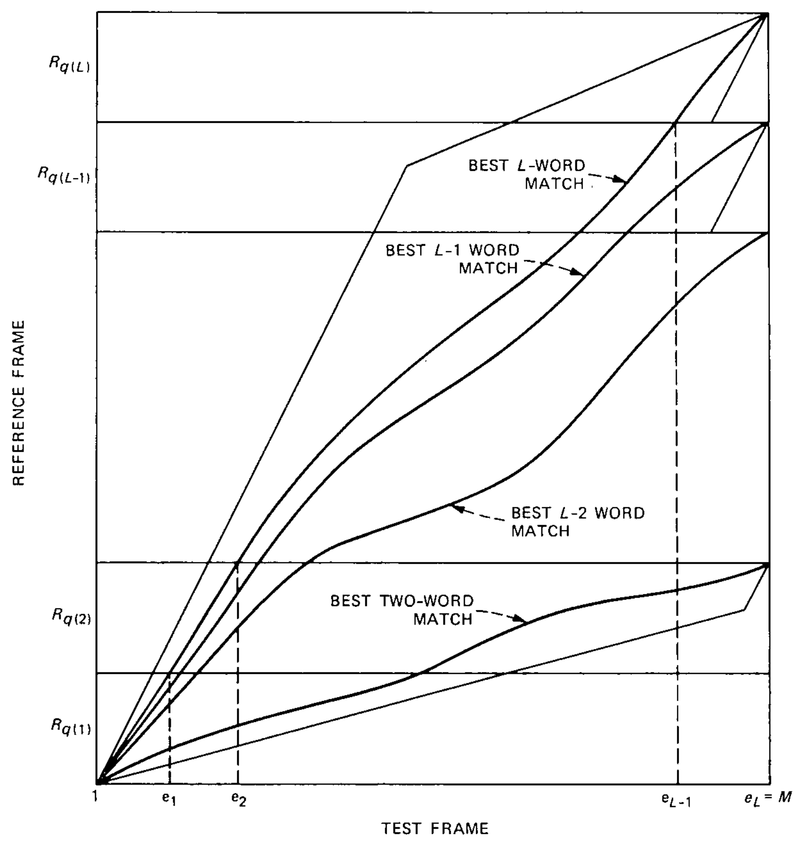

Fig. 2—Sequence of level-building DTW warps to provide best word sequences of several different lengths.

mentioned above, to show how we can use them to obtain improved speaker-independent word reference patterns.

Generally, the single word reference patterns used in the matching procedure of Figs. 1 and 2 are chosen as isolated occurrences of each vocabulary word (often obtained by some form of robust training procedure[13]). This form of training is adequate as long as the rate of articulation of the spoken connected word strings is not too high (e.g., typically fewer than 150 words per minute). However, for high rates of articulation, problems occur in the matching due to the gross differences between isolated words and those in fluent strings.

One solution to the problem of high rate of articulation is to use reference tokens extracted from connected word strings to supplement the isolated word reference tokens.[14] Such embedded training tokens are extracted from known training strings and can be used in a modified form of the robust training procedure to give robust embedded training tokens for each word of the vocabulary. Such techniques have been applied to the problem of connected digit recognition in both the speaker-trained mode,[14] and in the speaker-independent mode,[15] with very good success. The connected digits case was a natural first application since the number of environments in which each digit occurs is strictly limited (i.e., there are at most 10 predecessor digits and 10 following digits; furthermore, there is great similarity in many of these combinations).

Another natural application of connected word recognition is the case of recognition of connected letters for retrieving a name from a fixed directory of names.[16,17] Early experimentation with this system indicated that the task was a viable one (if the rate of articulation was not too high), based on isolated reference patterns for each letter.[18] In this paper we extend the embedded training procedure to handle the case of connected letters, and show how the resulting embedded letter templates can be used to improve recognition performance in a name retrieval task. The systems we used are speaker-trained ones; however, previous studies indicate that our results can readily be extended to the speaker-independent case.[15]

The organization of this paper is as follows. In Section II we review the structure of the overall connected letter recognition system and show how it can be used to retrieve the "best" matching name in a fixed directory of names. In Section III we discuss an evaluation of the overall connected letter, directory listing retrieval system, running in a speaker-dependent mode. In Section IV we discuss the results and give an analysis of the errors. Finally, in Section V we summarize our results and our main conclusions.

II. THE CONNECTED LETTER RECOGNITION SYSTEM

Figure 3 is a block diagram of the connected letter recognition system, as it used in the directory listing retrieval application. The system operates as follows. A user spells the last name of the person for whom directory information is desired as a connected sequence of spoken letters, followed by a brief pause, followed by the initials (again as a connected sequence). A conventional endpoint detector[19] finds the beginning and ending of each of the two spoken strings. An example of such endpoint detection is given in Fig. 4, where the dashed lines indicate the speech endpoints. An 8-pole Linear Predictive Coefficient (LPC) analysis is performed on each frame of both the spoken last name and the initials, where the analysis frame size is 45 ms and consecutive frames are spaced 15 ms apart. For both the spoken last name and the initials, a level-building Dynamic Time Warping (DTW) fit to a set of letter classes is made (based on letter reference patterns)

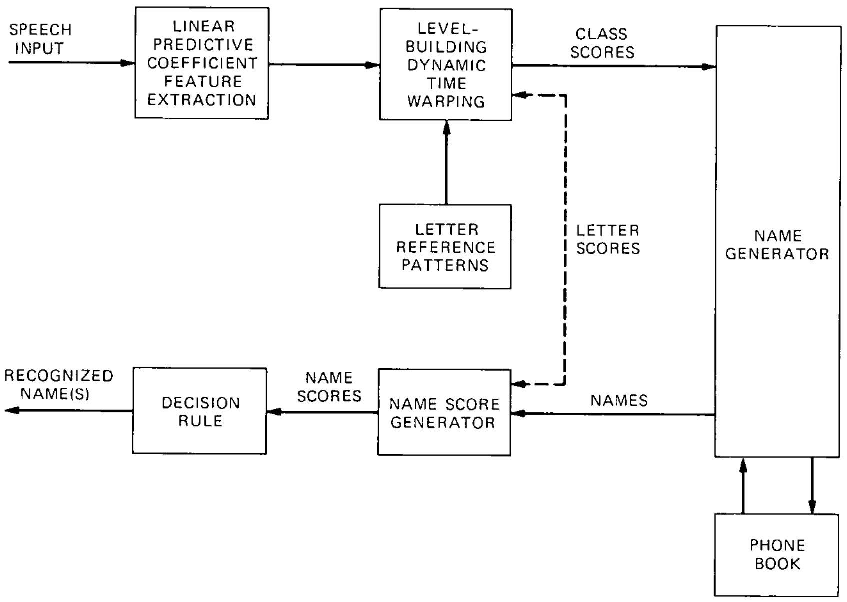

Fig. 3—Automatic directory listing retrieval system based on connected letter spelling of names.

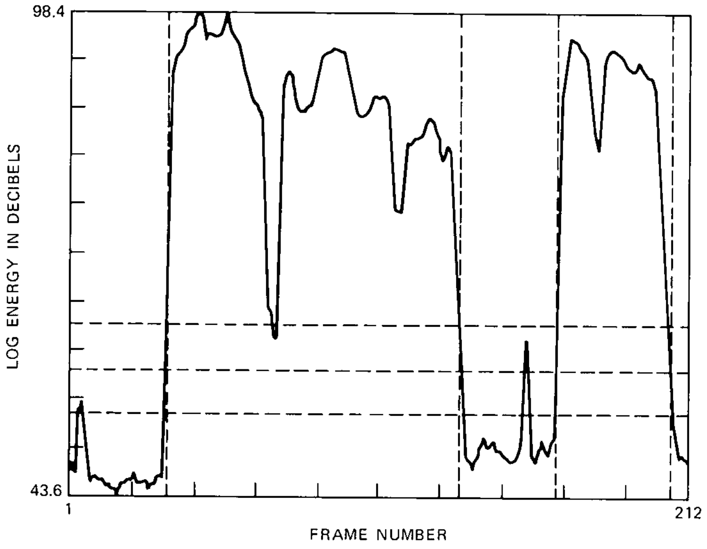

Fig. 4—Intensity contour for spelled name. The first set of vertical dashed lines delimits the beginning and end of the spoken last name; the second set of vertical dashed lines delimits the initials. The horizontal dashed lines indicate energy thresholds from which the beginning and ending frames are found.

and both the individual letter scores and the classes scores are saved. Last name class scores for all possible last name classes are generated and sorted by distance. A name generator sequentially goes through the sorted class list and generates all valid names within the class (i.e., those stored in the phone book). A name score generator uses the letter scores to give a total name score for each name within each class. Name scores are sorted in a list according to total name distance. Classes are searched until the best possible name score exceeds a specified threshold (related to the best name distance achieved so far). A list of the best name scores is then returned and the name recognized is the one at the top of the list.

In the remainder of this section we briefly describe the individual steps used to recognize the name. First we describe some characteristics of the letter classes and the phone book and then discuss the way in which we extract the embedded letter reference patterns for each talker. Next we review the level-building DTW algorithm and follow with a discussion of the name generation, name scoring, and decision rules.

2.1 Classification of letters into letter classes

The concept of blocking letters into letter classes, for purposes of speech recognition, was introduced by Aldefeld et al.[20] for the connected letter recognition application. The basic idea is that highly accurate recognition of spelled letters (over dialed-up telephone lines) cannot be achieved. Hence it is preferable to combine highly confusable letters into letter classes, perform recognition on letter classes, and decode the letter classes into actual directory names by searching a directory sorted by letter class combinations. Name scores are generated on the basis of individual letter scores (which are also generated in the recognition phase).

In particular, the 26 letters of the alphabet were assigned to 3 letter classes as shown in Table I. (A fourth class, class 0, contains the space

Table I—Assignment of letters into letter classes

Letter Class			
0	1	2	3
ƀ	B	A	F
	C	H	L
	D	I	M
	E	J	N
	G	K	Q
	P	O	R
	T	W	S
	V	Y	U
	Z		X

character, b̸). Class 1 contains the /EE/ letters, whereas classes 2 and 3 are a partitioning of the remaining 17 letters into two disjoint sets with minimal interclass confusion.[21] We denote the total number of classes as C.

For each name in the directory a set of I indices for the last name are defined. These indices define the letter class of each letter of the last name. Hence the names Rabiner and Wilpon would be represented by the indices:

NAME:	RABINER	WILPON
CLASS:	3 2 1 2 3 1 3,	2 2 3 1 2 3.

If we restrict ourselves to using $I = 6$ indices for the last name, and we adopt the convention that we use the character b̸ to pad out last names of fewer than six letters, then we have a total of

$3^6 =$	729	Classes with 6 letters
$3^5 =$	243	Classes with 5 letters
$3^4 =$	81	Classes with 4 letters
$3^3 =$	27	Classes with 3 letters
$3^2 =$	9	Classes with 2 letters
$3^1 =$	3	Classes with 1 letter
	1092	Total Classes.

After sorting an AT&T Bell Laboratories directory of 18,210 names according to the letter class assignment, a total of 1053 of the 1092 classes actually had one or more names assigned to it. Hence, coding of the last name to six indices is an efficient representation in terms of usage of possible letter classes.

2.2 Extraction of embedded letter patterns

The set of letter reference patterns for each talker consisted of three robust tokens of each letter, obtained as follows. An isolated robust token was obtained in the conventional manner, i.e., the talker spoke the letter repeatedly until two tokens were sufficiently similar (at a small enough distance) that they could be averaged.[13] Two embedded robust tokens of each letter were obtained by having the talker speak specified three-letter strings, extracting the middle letter via DTW alignment, and then using the standard robust training procedure on the embedded letters.[14] One of the embedded robust tokens was extracted from three-letter strings with minimal coarticulation between letters at the boundary. The other embedded robust token was extracted from three-letter strings with strong coarticulation between letters at the boundary.

Table II—Training sequences used for extraction of robust embedded letters for A, E, S, and W

Letter			
A	E	S	W
Noncoarticulated Sequences			
FAC	FEK	FSR	SWP
HAP	SEQ	XSW	XWT
XAT	XEK	HSL	FWK
SAQ	HEK	XSN	HWQ
FAP	FEQ	FSR	FWC
HAQ	XEQ	HSW	SWK
XAC	HEK	XSR	HWQ
Coarticulated Sequences			
RAL	WEL	WSC	WWR
MAN	NES	NSK	LWY
PAY	MEL	LSP	MWW
RAW	YEN	RSQ	LWR
ZAL	WER	LST	MWN
JAR	MEN	MSP	NWF
DAN	DEY	YSK	TWN

By way of example, Table II shows the three-letter training sequences used to obtain the robust tokens for the letters A, E, S, and W. The first seven sequences for each letter represent noncoarticulated strings; the next seven sequences for each letter represent coarticulated strings. The talker was only required to speak as many strings as required for obtaining a robust token of the letter. In theory as few as two strings could be adequate for this purpose; in practice it usually took four or five strings to give a pair of consistent embedded tokens. This is due to the high degree of variability of spoken letters in spelled strings.

An important point should be made about the training sequences. In theory the three-letter sequences were obtained by randomly selecting an initial and a final letter from a set of candidates based on the manner of production of the middle letter (the one being extracted) at the beginning and end of the word. In fact we found that the conventional vowels, namely A, E, I, O, and U, could not be used in either initial or final position in the training strings. Such combinations almost always led to extremely poor alignment paths for determining the embedded letter boundaries. As such, these letters were eliminated from consideration for use in the embedded training procedure.

The results of the training (which typically required about 30 minutes per talker) were a set of three reference patterns for each letter or a total of 78 reference patterns for the 26 letters.

Table III—Example of use of LB DTW algorithm in recognizing a spoken name

(a) Level-Building Distance Scores for Last Name and Initial Classes

Name:	ZBOYAN	AM
Class:	1 1 2 2 2 3	2 3

Last Name Class Scores:

	Class	Distance
$l = 1, 2, 3, 4, 5$ No matches found		
$l = 6$		
	112213	0.214
	112223	0.224
	212213	0.246

Initial Class Scores:

	Class	Distance
$l = 1$		
	2	0.449
$l = 2$		
	23	0.232
	13	0.267

(b) Overall Name Distance Scores

Class	Name	Overall Distance
112223	ZBOYAN AM	0.226
112223	ZBOYAN DL	0.235

2.3 Level-building recognition procedure

The recognition procedure is based on using the Level-Building (LB) DTW algorithm on strings of letter classes by using all 26 letters at each level but considering them only as different class templates. That is, different letters in the same letter class are considered as different templates for their common letter class. In the LB implementation we keep track of the C best (class) candidates at each level and use the standard LB traceback algorithm[9] to generate a name class score for each of the 1053 possible last name classes.

By way of example, Table III illustrates the application of the LB DTW algorithm on the spoken name ZBOYAN AM. We use the $C = 3$ letter classes shown in Table I. The first step is to generate last name class scores for each of the 1053 last name classes and to order these scores by distance. The results are shown in Table IIIa for the top three classes. Since a six-letter last name was spelled, the top three last name class distance scores are for three-letter last names. The last name class corresponding to the spoken name is not the best class, but instead has the second best distance score.

The next step is to generate initial scores for all possible sets of one or two initials. For both the last name and the initials, the LB algorithm keeps track of the best individual letter scores at each level. This requires a reasonable amount of storage but leads to a very efficient procedure for generating name scores. To generate a name score, one merely backtracks the individual letter scores (for both last name and initials) from the appropriate memory stacks, and a total name score is generated as

$$D_{\text{NAME}} = \frac{D_{LN} \cdot L_{LN} + D_I \cdot L_I}{L_{LN} + L_I}, \tag{1}$$

where D_{LN} and D_I are the normalized distances for the last name and initials, and L_{LN} and L_I are the number of letters in the last name and initials.

2.4 Stopping criteria

As we discussed above, the process of recognizing a name from spoken spelled letters consists of:

1. Running the LB on the last name and initials.
2. Generating all possible last name class scores.
3. Sorting the list of last name class scores.
4. Examining the sorted last name class score list sequentially and generating names and name scores for all names in each class that is examined. Name scores are sorted directly into a name score list.
5. Continuing this procedure until a stopping criterion is satisfied. The stopping criterion is that the best possible name score for a given class exceeds the best actual name score (based on previously checked names) by a given threshold. The best possible name score for a given class is given by

$$\hat{D}_C = \frac{D_{LNC} \cdot L_{LNC} + \tilde{D}_I \cdot \tilde{L}_I}{L_{LNC} + \tilde{L}_I}, \tag{2}$$

where D_{LNC} is the last name class score for the class being examined, L_{LNC} is the number of letters in the last name class score, $\tilde{D}_I$ is the best possible initials score (as determined by the LB output on the initials), and $\tilde{L}_I$ is the number of initials corresponding to the best initials score. The threshold used (in our simulation) for the stopping criterion was the value 0.06.

6. Once the stopping criterion was satisfied, the system returned the sorted list of names scores, and the recognized name was chosen as the one with the smallest distance.

Table IIIb shows the results of running steps 4 and 5 on the sequence of last name and initial scores used to generate the data of Table IIIa. Only two names had distance scores within the threshold, the best score corresponded to the actual spoken name in this case.

III. EXPERIMENTAL EVALUATION

To evaluate the performance of the directory listing retrieval system described in Section II, four talkers (three male, one female) each trained the recognizer using the robust training procedure to give the isolated and embedded templates for each letter. The four talkers were all experienced users of speech recognition systems. These same talkers each provided a test set of 50 randomly chosen names from the 18,210-name directory (the set of 50 names was different for each talker). Each name in the test set was spoken as a sequence of connected letters for the last name, followed by a pause, followed by a sequence of connected letters for the initials. The talkers spoke each name at a normal rate. All recordings were made over local dialed-up telephone lines. The average talking rates for the four talkers are shown in Table IV. The average rates for the last name vary from 189 words per minute (wpm) to 218 wpm; for the initials the rates vary from 140 to 167 wpm. Thus, the names were spoken at very fast rates of articulation.

IV. RECOGNITION RESULTS—SPEAKER-TRAINED CASE

The directory listing retrieval system of Section II was run on the 200 names by the four talkers in a speaker-dependent mode. The LB parameters (see Ref. 9 for a complete description of these parameters) were set to:

1. ϵ = width of DTW search region = 99
2. M_T = multiplier for interlevel scores = 2.2
3. δ_{END} = search region at end of string = 4
4. δ_{R1} = number of frames that can be skipped at beginning of reference template = variable
5. δ_{R2} = number of frames that can be skipped at end of reference template = variable
6. Inserted silence at beginning or end of reference template = variable.

The values for δ_{R1} and δ_{R2} were made variable with the templates—i.e., different values were used for the isolated pattern than for each

Table IV—Talking rates (wpm) for the four talkers

Talker	Last Name Rate (wpm)	Initials Rate (wpm)
1	189	140
2	210	142
3	210	159
4	218	167

of the embedded patterns. In particular we considered three sets of δ_{R1} and δ_{R2} values, namely:

1. $\delta_{R1} = (0, 0, 0)$, $\delta_{R2} = (4, 0, 0)$
2. $\delta_{R1} = (4, 0, 0)$, $\delta_{R2} = (6, 0, 0)$
3. $\delta_{R1} = (4, 2, 0)$, $\delta_{R2} = (6, 3, 0)$,

where the first value is for the isolated pattern, the second value is for the noncoarticulated pattern, and the third value is for the coarticulated pattern. The first set of values was the optimal one for speaker-independent patterns for digits;[15] the second set of values was optimal for speaker-dependent patterns for digits;[14] the third set of values is a compromise that provides some template shortening for the noncoarticulated reference patterns.

The inserted silence parameter is the number of frames of silence put at either the beginning or end of templates to reflect the presence of initial or final stops in the word. The letters of the alphabet for which initial silence was used were b, d, g; and p, t, k, and q. None of the letters used final silence insertion. Based on some preliminary experimentation, three sets of silence values were used, namely:

1. 0 for {b, d, g}, 0 for {p, t, k, q}
2. 2 for {b, d, g}, 3 for {p, t, k, q}
3. 4 for {b, d, g}, 6 for {p, t, k, q},

where the parameter values are in terms of frames; hence, four frames corresponds to 60 ms of silence insertion.

A series of recognition tests were performed in which δ_{R1} and δ_{R2} were varied along with the silence insertion variable. For each of these tests, name recognition accuracy was measured for three sets of speaker-dependent reference templates,* namely:

1. IS = Isolated templates alone
2. IS $\oplus$ NC = Isolated plus noncoarticulated embedded templates
3. IS $\oplus$ NC $\oplus$ CO = Isolated plus both types of embedded templates.

Results for each of the nine sets of variables and for the three template sets are given in Table V. The results given in this table show the following:

1. For the IS template set, values of (0, 0, 0) and (4, 0, 0) for δ_{R1} and δ_{R2} lead to extremely poor performance. For all other choices of parameter values, the system performance is essentially identical at 86 percent $\pm$ 1 percent. These results point out the necessity of being able to skip some reference frames at the beginning of each template.

2. For the IS $\oplus$ NC template set, there is only a small variation in performance (from a low of 91 percent to a high of 94 percent) as the

* Runs were also made with other combinations of reference patterns (e.g., NC templates alone), but the three sets discussed below provided the most interesting and informative results.

Table V—Recognition accuracy as a function of silence parameters

	δ_{R1} and δ_{R2} Values		
	(0, 0, 0)	(4, 0, 0)	(4, 2, 0)
Template Set	(4, 0, 0)	(6, 0, 0)	(6, 3, 0)
(a) Values of 0 for {b, d, g} and 0 for {p, t, k, q}			
IS	53.5	86	86
IS ⊕ NC	92	93	94
IS ⊕ NS ⊕ CO	93	94	91.5
(b) Values of 2 for {b, d, g}, and 3 for {p, t, k, q}			
IS	50.5	87	87
IS ⊕ NC	92.5	92.5	91
IS ⊕ NC ⊕ CO	94	93.5	95
(c) Values of 4 for {b, d, g} and 6 for {p, t, k, q}			
IS	45.5	85	85
IS ⊕ NC	91	93	92
IS ⊕ NC ⊕ CO	91.5	94	94

δ_{R1}, δ_{R2}, and silence parameters are varied. When we compare the performance to that obtained from the IS template set, we can see that the inclusion of the embedded NC templates leads to improved performance, as well as to insensitivity to the exact parameter values of the variable system parameters.

3. For the IS ⊕ NC ⊕ CO template set, only very slight improvements in performance are obtained over that of the IS ⊕ NC template set, and these improvements do not occur for all values of the variables. The highest accuracy obtained is 95 percent; however, four different sets of parameter values yield 94-percent name accuracy.

A further analysis of the results for the best parameter set is given in Table VI. Included in this table are the individual recognition accuracies as a function of candidate position (top β candidates, β = 1, 2, 3, 4, 5) for each talker (along with the average), as well as two measures of the amount of searching performed to find the best name. The search measures are $\bar{C}_s$, the average number of last name classes searched, and $\bar{N}_s$, the average number of names whose distance score was evaluated. The results in Table VI show that two of the talkers performed well using IS templates, but the other two talkers performed very poorly. For these other two talkers the inclusion of embedded templates led to improvements in performance. It can also be seen that a 2- to 3-percent improvement in accuracy can be obtained by considering the second candidate position scores—i.e., about 2 to 3 percent of the time the correct name is in second position. Such cases are typically names with slight (within letter class) errors in the initials.

By examining the search statistics we see that the average search

Table VI—Percentage of individual recognition accuracies as a function of candidate position and search statistics for each talker for three template sets using $\delta_{R1} = (4, 2, 0)$ and $\delta_{R2} = (6, 3, 0)$, and silence for $\{b, d, g\} = 2$ and $\{p, t, k, q\} = 3$

	Candidate Position						
Talker	1	2	3	4	5	$\bar{C}_s$	$\bar{N}_s$
(a) Results Using IS Template Set							
1	76	82	84	84	86	114	1976
2	80	82	82	86	86	139	2553
3	94	96	96	96	96	36	671
4	98	98	98	100	100	37	665
Average	87	89.5	90	91.5	92	81.5	1466
(b) Results Using IS ⊕ NC Template Set							
1	82	88	88	88	88	111	2086
2	94	96	98	98	98	113	2308
3	94	98	98	98	98	10	203
4	94	94	94	96	96	40	675
Average	91	94	94.5	95	95	68.5	1318
(c) Results Using IS ⊕ NC ⊕ CO Template Set							
1	94	98	98	98	98	56	1011
2	92	94	94	94	94	70	1416
3	96	98	98	98	98	19	352
4	98	98	98	100	100	16	296
Average	95	97	97	97.5	97.5	40.3	769

time for the three-template-per-word set is about one-half that of the IS template set. Hence, the embedded templates yield considerably more accurate name classes than that obtained from the IS templates alone.

An analysis of the 10 name errors (in first candidate position) for the IS ⊕ NC ⊕ CO template set of Table VIc shows the following:

1. Four of the errors were due to errors in initials of people with the same last name. In all cases these errors were within letter class errors, i.e., JR confused with AR.

2. Four of the errors were due to a known flaw in the level-building algorithm[22] in which the second best path to a given frame need not be the optimal second best path. Such cases could be potentially corrected at the expense of greatly increased computation in the LB algorithm.

3. Two of the errors were names that could not be matched by the individual letter patterns. Such cases were highly coarticulated letter sequences whose matches were extremely poor in the LB algorithm.

4.1 Recognition results—speaker-independent case

The same set of 200 names was used as a test of the connected letter directory listing retrieval system in a speaker-independent mode. For

Table VII—Percentage of individual recognition accuracies as a function of candidate position and search statistics for each talker for the speaker-independent case

	Candidate Position						
Talker	1	2	3	4	5	$\bar{C}_s$	$\bar{N}_s$
1	88	90	94	94	96	29	473
2	70	80	82	88	88	185	3137
3	84	90	90	92	92	110	2097
4	66	74	74	76	78	174	3011
Average	77	83.5	84	87	88	124.5	2179

this test the letter reference patterns were a set of 12 isolated templates per letter, the templates having been extracted from a clustering analysis of isolated occurrences of each letter by 100 talkers (50 male, 50 female).

The results of the recognition test are given in Table VII, which gives recognition accuracy as a function of candidate position for each of the four talkers (as well as the average), and the average search statistics. Overall, we can see that degraded performance results from the use of only isolated templates. An average name recognition accuracy of 77 percent is achieved, as opposed to the 95-percent accuracy in the speaker-dependent case. The inherent system difficulties are illustrated in the average search statistics, which show that it took about 124 class evaluations and 2179 name evaluations to find the best name—a factor of three to one greater than required for the speaker-trained case.

An analysis of the 46 name errors (out of the 200 names) shows:

1. Nine errors where only the initials were incorrect.

2. Twenty-two errors in which the last name was in error—i.e., a match to an incorrect last name was better than the match to the correct last name. In such cases the match to the initials was not able to correct the errors.

3. Fifteen errors in which the correct name could not be matched because of the LB flaw discussed earlier, or because the rate of articulation of the letters exceeded the rate at which a match could be achieved from isolated letter templates.

V. DISCUSSION

To get some perspective on the relevance of the results given in Section III, we must compare the current system performance against that achieved in earlier implementations subject to the experimental constraints of small sample populations (i.e., the use of only four test talkers). In the most relevant comparison, Myers and Rabiner[18] studied a similar system in which a fixed set of 50 names was used as the test by each of four talkers (different from those used here). Using isolated templates alone, recognition accuracies of 90.5 percent and 87.5 percent were achieved in the speaker-dependent and speaker-independent modes, respectively, for normally spoken names. Earlier work by Aldefeld et al. has shown that the 50 chosen names tended to give an overbound on true system performance as there was no adequate representation of the common problems in name spelling (e.g., multiple names with differing initials, names differing in a single letter, etc.).[21] Hence the recognition accuracies of 87 percent and 77 percent for the case of isolated templates in the speaker-dependent and speaker-independent modes are comparable to those reported earlier.

The recognition performance using embedded training patterns in the speaker-dependent case indicates an improvement in recognition accuracy along with a significant reduction in search time to find the best name. Hence we conclude that, as in the connected digits recognition task, the use of embedded word training can and does enhance the recognition system performance.

At this point it is natural to ask "Where do we go from here to improve the performance of a connected word recognition task based on pattern matching techniques?" There are two obvious directions for making improvements.

First, the embedded training procedure should be able to provide more than a single embedded pattern when necessary. For example, it was clear that some letters are more influenced by context than others. In such cases the resulting robust embedded pattern was often a poor representative of the letter, and did not yield good matching scores in real names. Using two or more embedded tokens obtained via some clustering procedure would have aided greatly in such cases.

A second obvious direction for improvement is to make no decision on names in which two or more name candidates are within some reasonable distance of the best name score. In such cases (mostly involving initial errors) additional information should be requested from the talker to help resolve ambiguity. Such a strategy was used with great success by Aldefeld et al.[21] in a practical implementation of this system with isolated letter input.

VI. SUMMARY

We have shown how a practical directory listing retrieval system could be implemented on the basis of connected letter name spelling. Our results indicate that improved recognition performance can be obtained when combining embedded letter patterns (suitably extracted from three-letter strings) with the standard isolated letter patterns to form an enhanced letter reference set. Using this combined set of references, improvements in name accuracy of 8 percent and reductions in search time of a factor of two result, when the system is tested in a speaker-trained mode using dialed-up telephone line recordings.

VII. ACKNOWLEDGMENT

The authors thank Tom Gruenenfelder and an anonymous *AT&T Bell Laboratories Technical Journal* reviewer for their comments and criticisms of this paper.

REFERENCES

1. F. Itakura, "Minimum Prediction Residual Principal Applied to Speech Recognition," IEEE Trans. Acoustics, Speech, and Signal Processing, *ASSP-23* (February 1975), pp. 67–72.
2. L. R. Rabiner and S. E. Levinson, "Isolated and Connected Word Recognition—Theory and Selected Applications," IEEE Trans. Commun., *COM-29,* No. 5 (May 1981), pp. 621–59.
3. W. A. Lea, ed., *Trends in Speech Recognition,* Englewood Cliffs, NJ: Prentice-Hall, 1980.
4. T. B. Martin, "Practical Applications of Voice Input to Machines," Proc. IEEE, *64* (April 1976), pp. 487–501.
5. G. R. Doddington and T. B. Schalk, "Speech Recognition: Turning Theory to Practice," IEEE Spectrum, *18,* No. 9 (September 1981), pp. 26–32.
6. S. Moshier, "Talker Independent Speech Recognition in Commercial Environments," Speech Commun. Papers 97th ASA Meeting, (June 1979), pp. 551–3.
7. W. A. Lea, "Selecting the Best Speech Recognizer for the Job," Speech Technology, *1,* No. 4 (January/February 1983), pp. 10–29.
8. H. Sakoe, "Two Level DP-Matching—A Dynamic Programming Based Pattern Matching Algorithm for Connected Word Recognition," IEEE Trans. Acoustics, Speech, and Signal Processing, *ASSP-27* (December 1979), pp. 588–95.
9. C. S. Myers and L. R. Rabiner, "Connected Digit Recognition Using a Level Building DTW Algorithm," IEEE Trans. Acoustics, Speech, and Signal Processing, *ASSP-29,* No. 3 (June 1981), pp. 351–63.
10. J. S. Bridle, M. D. Brown, and R. M. Chamberlain, "An Algorithm for Connected Word Recognition," Automatic Speech Analysis and Recognition, ed. J. P. Haton, Dordrecht, Holland; D. Rydell Publishing Co., 1982, pp. 191–204.
11. J. L. Gauvain and J. Mariani, "A Method for Connected Word Recognition and Word Spotting on a Microprocessor," Proc. 1982 ICASSP (May 1982), pp. 891–4.
12. M. H. Kuhn and H. H. Tomaschewski, "Improvements in Isolated Word Recognition," IEEE Trans. Acoustics, Speech, and Signal Processing, *ASSP-31,* No. 1 (February 1983), pp. 157–67.
13. L. R. Rabiner and J. G. Wilpon, "A Simplified Robust Training Procedure for Speaker Trained, Isolated Word Recognition Systems," J. Acoust. Soc. Amer., *68,* No. 5 (November 1980), pp. 1271–6.
14. L. R. Rabiner, A. Bergh, and J. G. Wilpon, "An Improved Training Procedure for Connected-Digit Recognition," B.S.T.J., *61,* No. 6 (July–August 1982), pp. 981–1001, pp. 157–67.
15. L. R. Rabiner, J. G. Wilpon, A. M. Quinn, and S. G. Terrace, "On the Application of Embedded Digit Training to Speaker Independent, Connected Digit Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-32* (1984).
16. A. E. Rosenberg and C. E. Schmidt, "Automatic Recognition of Spoken Spelled Names for Obtaining Directory Listing," B.S.T.J., *58,* No. 8 (October 1979), pp. 1797–823.
17. A. E. Rosenberg, L. R. Rabiner, and J. G. Wilpon, "Recognition of Spoken Spelled Names for Directory Assistance Using Speaker-Independent Templates," B.S.T.J., *59,* No. 4 (April 1980), pp. 571–92.
18. C. S. Myers and L. R. Rabiner, "An Automated Directory Listing Retrieval System Based on Recognition of Connected Letter Strings," J. Acoust. Soc. Am., *71,* No. 3 (March 1982), pp. 716–27.

19. L. F. Lamel, L. R. Rabiner, A. E. Rosenberg, and J. G. Wilpon, "An Improved Endpoint Detector for Isolated Word Recognition," IEEE Trans. Acoustics, Speech, and Signal Processing, *ASSP-29,* No. 4 (August 1981), pp. 777–85.
20. B. Aldefeld, S. E. Levinson, and T. G. Szymanski, "A Minimum Distance Search Technique and Its Application to Automatic Directory Assistance," B.S.T.J., *59* (October 1980), pp. 1343–56.
21. B. Aldefeld, L. R. Rabiner, A. E. Rosenberg, and J. G. Wilpon, "Automated Directory Listing Retrieval System Based on Isolated Word Recognition," Proc. IEEE, *68* (November 1980), pp. 1364–79.
22. C. S. Myers and L. R. Rabiner, "A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word Recognition," B.S.T.J., *60,* No. 7 (September 1981), pp. 1389–409.

AUTHORS

Lawrence R. Rabiner, S.B. and S.M., 1964, Ph.D., 1967 (Electrical Engineering), The Massachusetts Institute of Technology; Bell Laboratories, 1962—. Presently, Mr. Rabiner is engaged in research on speech communications and digital signal processing techniques. He is coauthor of *Theory and Application of Digital Signal Processing* (Prentice-Hall, 1975), *Digital Processing of Speech Signals* (Prentice-Hall, 1978), and *Multirate Digital Signal Processing* (Prentice-Hall, 1983). Member, National Academy of Engineering, Eta Kappa Nu, Sigma Xi, Tau Beta Pi. Fellow, Acoustical Society of America, IEEE.

Sandra G. Terrace, B.S. (summa cum laude, in Mathematics), 1975, Northeastern University, Boston, Massachusetts. Ms. Terrace works primarily from her home in Chelmsford, Massachusetts, where she and her husband own and operate a computer software consulting firm. She is a computer software consultant to the Acoustics Research Department of AT&T Bell Laboratories at Murray Hill, New Jersey. Since April 1980 she has been engaged in the development of speech-recognition and speech-processing algorithms with the CSPI MAP Array Processor. Programs have been developed using MAP Assembly languages and Data General FORTRAN to accomplish isolated and connected speech recognition, speech synthesis, and speech endpoint detection.

Jay G. Wilpon, B.S. and A.B. (cum laude) in Mathematics and Economics, respectively, 1977, Lafayette College, Easton, Pennsylvania; M.S. (Electrical Engineering/Computer Science), 1982, Stevens Institute of Technology, Hoboken, N.J.; AT&T Bell Laboratories, 1977—. Since June 1977 Mr. Wilpon has been with the Acoustics Research Department at AT&T Bell Laboratories, Murray Hill, N.J., where he is a Member of the Technical Staff. His interests include basic research in the field of speech recognition, training algorithms for speaker-dependent and speaker-independent recognizers, and speech endpoint detection.