Copyright © 1979 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 58, No. 8, October 1979
Printed in U.S.A.

Automatic Recognition of Spoken Spelled Names for Obtaining Directory Listings

By A. E. ROSENBERG and C. E. SCHMIDT

(Manuscript received June 9, 1978)

An automatic speaker-dependent word recognizer is used to accept strings of spelled letters spoken in isolation. The output of the recognizer is a set of best-candidate letters for each letter spoken in the string. Candidate strings forming spelled last names and initials are compared to name strings obtained from a telephone directory stored in a disk file. A systematic search is carried out to find matching entries in the directory. An evaluation has been carried out with ten talkers spelling lists of names extracted from an 18,000-entry telephone directory. Although the median acoustic error rate per spelled letter is approximately 20 percent, the median error rate in furnishing the requested directory entry is approximately 4 percent.

I. INTRODUCTION

Since July 1976, an automatic directory assistance system has been available at Bell Laboratories via a dialed-up connection which provides Bell Laboratories telephone directory information in response to *TOUCH-TONE*® dial inquiries.[1,2] The system has been implemented on a Data General Nova 800 laboratory computer and makes use of an ADPCM (adaptive differential pulse-code modulation) voice-response system,[3] which is interfaced to the computer, to provide instructions and the desired directory information to the calling customer. To use the system, the caller dials the appropriate telephone number and then, following a voice-response prompt, spells out on the *TOUCH-TONE* dial the last name and initials of the individual for whom directory information is desired. A caller's spelled inquiry on the *TOUCH-TONE* dial can be compared directly with the names in a *TOUCH-TONE* version of the telephone directory and a list of possible matches compiled. In approximately 75 percent of all inquiries for which matches are obtained, matches are unique. For the remaining 25 percent, multiple matches are obtained due to individuals whose names are spelled identically (at least with respect to those letters supplied in the inquiry) as well as spelling ambiguities arising from the fact that 3 or more letters are assigned to each dial button. Inquiries resulting in multiple matches are followed up by requests to the customer to supply additional information which can often resolve the ambiguity.

The *TOUCH-TONE* system has been demonstrated to have practical potential for automating customer inquiries to directory assistance. Nevertheless, mechanical intervention in the form of actuating the proper sequence of pushbuttons is required of the customer. Any such process mediating the customer's inquiry is likely to be accompanied by error. Moreover, spelling ambiguities arising from multiple letter assignments on the buttons result in a significant number of multiple matches. Although such matches are handled in a clever way, they require additional information to be extracted from the customer and prolong the inquiry.

Obviously the most direct and attractive mode for directory assistance inquiries is spoken utterances. It is also the most elusive for automation, since an automatic speech recognizer that could handle spoken inquiries in the most accommodating and efficient manner has yet to be devised. Moreover, any such device would have to be enormously complex compared with *TOUCH-TONE* inquiry systems. Nevertheless, practical systems for automatic speech recognition are currently available if enough restrictions are imposed. It is the purpose of this paper to demonstrate how such an existing speech recognizer might be incorporated into a system for voice-actuated directory assistance. The system does not represent the ultimate answer for voice-actuated inquiries; rather, it points towards a practical means given the current abilities of automatic speech recognizers.

Typically, the restrictions on contemporary automatic speech recognizers include utterances spoken in isolation by designated speakers from a limited vocabulary. Such a system has been implemented and evaluated at Bell Laboratories.[4,5] A block diagram of the system is shown in Fig. 1. The acoustic input signal is parameterized in terms of an eighth-order linear predictor coefficient (LPC) analysis. A description of the system operation is given in the next section.

Given the limited vocabulary restriction of contemporary word recognizers, it would not be possible to design a practical, automatic, directory assistance system in which spoken names are used to make requests for information. However, a vocabulary consisting of spelled letters has a manageable size, and spelling names to request directory information is a natural means of making such inquiries. In fact, requests to human directory-assistance operators often include spellings. One drawback, however, is that the spoken spelled alphabet is a

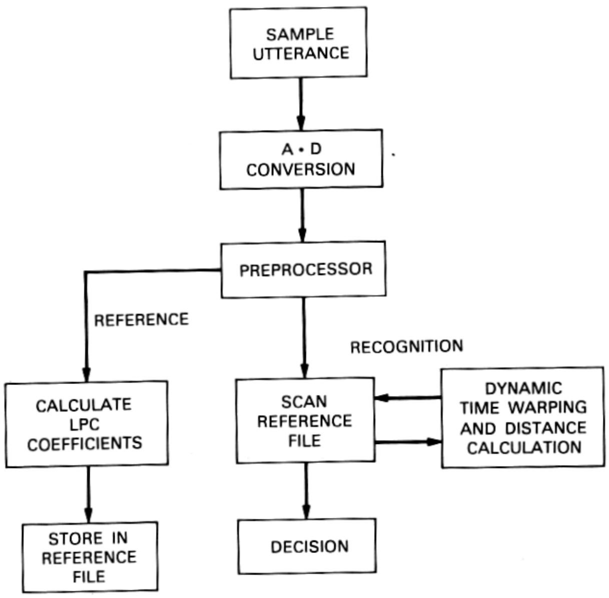


Fig. 1—Automatic word recognizer block diagram.

notoriously poor vocabulary for a word recognizer. This is because large groups of utterances within the vocabulary are easily confused because they have minimal acoustic distinctions. This difficulty can be alleviated by adopting modified recognition techniques. For example, the Itakura recognizer is essentially a template matcher and provides an overall distance or figure of merit for every template or reference pattern to which a sample pattern is compared. It is therefore possible to generalize the notion of recognition from a unique match to a set of possible matching candidates ordered by the overall distances provided by the recognizer. Even though there may be a significant probability that the correct word template is not the best matching candidate, the probability may be quite low that the correct word template is *not* found among, say, the five best matching candidates. Therefore, given a string of spoken letters as input to a recognizer, there is a high probability that the correct string of letters can be found among the candidates for recognition even though the recognizer is imperfect.

To find the correct matching string, the directory must be searched by systematically presenting to it candidate strings, starting with the most probable string as prescribed by the recognizer. In order for this kind of search to have a reasonable chance of obtaining the correct string, an additional condition must be satisfied—namely, the distribution of letter strings listed in the directory must be sparse compared with all possible letter strings. For example, these are 26^8 or approximately 2×10^{11} possible strings of length 8, but only a small fraction of these will be found in a telephone directory. The evaluation described in Section III shows that the conditions are, in fact, satisfied by the kind of combination of recognizer and directory search to be described. The constraints imposed by the distribution of directory strings are such that it is possible to relax conditions even further by ignoring one or more positions in the string and still find the correct matching string in the directory. This situation arises when the acoustic recognizer fails to find any letter candidate in a given position in the string.

II. DETAILS OF OPERATION

The operation of the Itakura automatic speaker-dependent word recognizer is outlined in the block diagram of Fig. 1. The input acoustic signal is digitized at a 6.667-kHz rate and processed through an endpoint detector to locate the beginning and end of the utterance. The preprocessing consists of an eighth-order autocorrelation analysis over a 45-ms Hamming window every 15 ms through the utterance. For reference patterns, the autocorrelation coefficients are converted frame-by-frame to LPC coefficients. Recognition consists of a matching process in which a sample input pattern of autocorrelation coefficients is compared with an ensemble of stored reference patterns previously established by the designated speaker. The comparison consists of a frame-by-frame scan of the sample patterns against each reference pattern. A distance metric (or measure of dissimilarity) is calculated and accumulated by a dynamic programming technique as the scan proceeds. An accumulated distance rejection threshold function is imposed such that, if the accumulated distance exceeds the threshold at any point during the scan against a particular reference pattern, the scan is aborted and restarted against a new reference pattern. The vocabulary item corresponding to the reference pattern with the lowest accumulated distance is designated as the recognized item. If every reference pattern is rejected, the result is said to be a rejection. For the purposes of the present system, the output of the recognizer is considered to be a *set* of candidates for recognition ordered by their associated accumulated distances. Since the set does not include rejected items, the number of candidates for each trial may vary from zero to the size of the vocabulary.

The vocabulary for the present system is shown in Table I. It consists of the spoken representations for each letter in the alphabet, the set of digits zero through nine, and three commands. Spoken digits are needed when additional information must be solicited from the customer to resolve ambiguities resulting from similar entries in the

Table I—Vocabulary

1. A	14. N	27. Stop
2. B	15. O	28. Error
3. C	16. P	29. Repeat
4. D	17. Q	30. Zero
5. E	18. R	31. One
6. F	19. S	32. Two
7. G	20. T	33. Three
8. H	21. U	34. Four
9. I	22. V	35. Five
10. J	23. W	36. Six
11. K	24. X	37. Seven
12. L	25. Y	38. Eight
13. M	26. Z	39. Nine

directory. Such information may be in the form of numerical information such as organization numbers in the Bell Laboratories directory or street addresses in a public directory. The "stop" command, analogous to the "*" button in the *TOUCH-TONE* system, is used to terminate strings of utterances. The "error" and "repeat" commands are used to correct input errors.

In its present configuration, the data base for the system is the approximately 18000-entry Bell Laboratories directory. Many of the directory access routines for this system have been modified from the *TOUCH-TONE* directory assistance system.[1,2]

The system is accessed using an ordinary telephone handset via a dialed-up connection through the PBX at Bell Laboratories in Murray Hill, N.J. Upon receipt of an audible cue, the customer spells out the name of the individual for whom directory information is required. The letters must be spoken distinctly, separated by intervals of at least 100 ms. As in the *TOUCH-TONE* system, the letters of the last name are spelled first, followed by a "stop" command, followed by the initials and a final "stop" command. The requirements for the number of last-name letters and initials are also the same as in the *TOUCH-TONE* system. The entire string of spelled letters is uttered at once with the frame-by-frame calculation and storage of autocorrelation coefficients accomplished in real time. Off-line, the utterances are segmented using the end-point detector. The comparison and matching process, which in the present configuration takes approximately six seconds per utterance, provides a maximum of five candidate letters for each spoken letter.*

The overall operation of the system is shown in a simplified flow chart in Fig. 2. The directory search starts with the construction of an

* The maximum figure of five candidates is chosen somewhat arbitrarily. If the maximum is set low, the likelihood of excluding the correct letter increases. If it is set high, the number of candidate strings and, hence, the search time increases. In an informal evaluation, five seemed to represent a reasonable compromise.

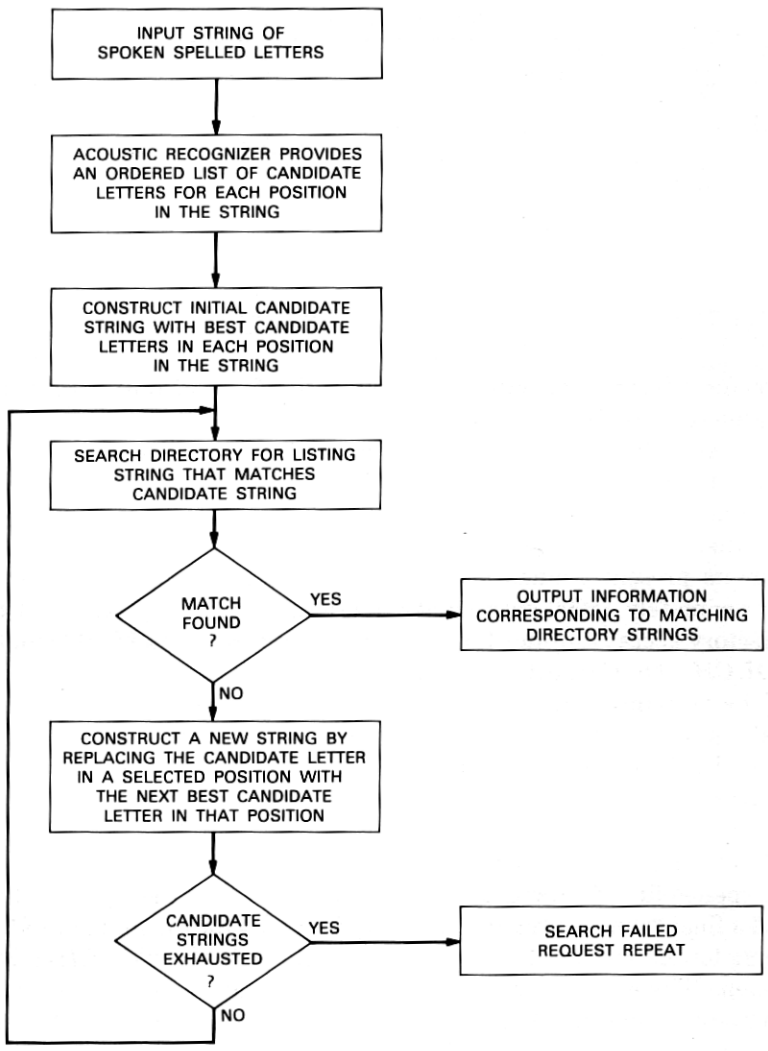

INPUT STRING OF SPOKEN SPELLED LETTERS
ACOUSTIC RECOGNIZER PROVIDES AN ORDERED LIST OF CANDIDATE LETTERS FOR EACH POSITION IN THE STRING
CONSTRUCT INITIAL CANDIDATE STRING WITH BEST CANDIDATE LETTERS IN EACH POSITION IN THE STRING
SEARCH DIRECTORY FOR LISTING STRING THAT MATCHES CANDIDATE STRING
MATCH FOUND ?
YES
OUTPUT INFORMATION CORRESPONDING TO MATCHING DIRECTORY STRINGS
NO
CONSTRUCT A NEW STRING BY REPLACING THE CANDIDATE LETTER IN A SELECTED POSITION WITH THE NEXT BEST CANDIDATE LETTER IN THAT POSITION
CANDIDATE STRINGS EXHAUSTED ?
YES
SEARCH FAILED REQUEST REPEAT
NO

Fig. 2—Simplified flow chart of the overall system.

initial candidate string. The string is composed of the best candidate letters supplied by the acoustic recognizer for each letter position in the string of last-name letters and initials.* If one or more matching directory listing strings are found, the directory information for these

* In order to simplify the system description, it is assumed that there is no distinction between last-name letters and initials for candidate and listing strings. In actual practice, initials are handled somewhat differently than last-name letters. It is also assumed that the "stop" command is recognized without error, which is very nearly true in practice.

listings are output to the customer. If no match is found, a new candidate string is constructed by replacing the candidate letter in a selected position by the next best candidate letter in that position. The directory search is then restarted. If all candidate strings are exhausted, the search has failed and the request must be repeated.

The process of searching for a directory entry which matches an input string using the array of candidate letters provided by the recognizer can be separated into two operations. First, a suitable sequence of candidate strings must be constructed. Second, the directory must be probed for the existence of a match to each such candidate string. The first operation is more interesting and critical in the context of this investigation than the second. An efficient search technique serves to speed up the process and assumes greater importance as the size of the directory increases. But the technique used to construct sequences of candidate strings significantly affects both the accuracy and the efficiency of the overall process.

The search technique can be described very simply. Advantage is taken of the linear, alphabetical arrangement of the directory, grouping the listings into "buckets"[6] containing 128 consecutive listings each. A table stored in memory indexes each 128th listing in the directory by its first three letters. To probe a given candidate string, a search range consisting of one or more "buckets" is selected keyed to the string in the index table closest to the candidate string. The range is chosen to ensure that a matching directory string, if it exists, is included within the range. The candidate string is compared with each directory string within the selected range.

The more critical operation, as indicated above, is the technique used to construct a sequence of candidate strings. It is useful to introduce the technique which is actually used in this study with a discussion of a general approach. As shown in Fig. 3, we can model the entire process as follows. The directory can be represented as a finite-state Markov process generating strings of letters forming names which are spelled by the speaker and processed by the recognizer. The speaker and recognizer together are considered a discrete memoryless channel. A directory entry is a sequence of characters $\cdots *, *, u_1, u_2, \cdots, u_N, *, *, \cdots$, where * represents a blank or space and u_k represents a character $A, B, C, \cdots, Z$ or *. The output of the directory model at position k in the string is taken to be

$$x_k = (u_k, u_{k-1}, \cdots u_{k-\nu}), \tag{1}$$

where it is supposed that the present state of the directory x_k is a function of the present and ν preceding characters. The observed output of the recognizer is a series of measurements $z_1, z_2, \cdots z_k, \cdots z_N$. We suppose that the directory is represented by state transition

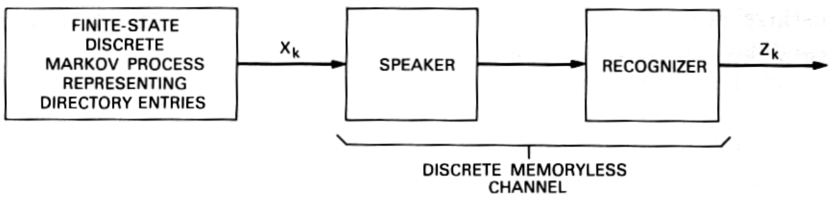

FINITE-STATE DISCRETE MARKOV PROCESS REPRESENTING DIRECTORY ENTRIES
x_k
SPEAKER
RECOGNIZER
z_k
DISCRETE MEMORYLESS CHANNEL

Fig. 3—Overall system model as a Markov process—memoryless channel.

probabilities $P(x_k/x_{k-1})$ and the speaker-recognizer combination by conditional probabilities $P(z_k/x_k)$. Since the speaker-recognizer combination is considered memoryless, we have

$$P(z_k/x_k) = P(z_k/u_k). \tag{2}$$

Given an observed sequence $\{z\}$, the problem is to reconstruct the input sequence $\{u\}$. This problem has been considered in many different forms, especially in coding theory studies.[7-9] A solution can be obtained by a maximum *a posteriori* probability estimation of $\{x\}$ or, equivalently, $\{u\}$, given $\{z\}$. It can be shown that solving the problem is equivalent to finding the shortest path through a graph which can be obtained by a dynamic programming technique.[7,9] The solution is provided by the Viterbi algorithm[3,10] stated as follows. Let

$$\lambda(x_k, x_{k-1}) = -\ln P(x_k/x_{k-1}) - \ln P(z_k/u_k). \tag{3}$$

For each k, compute

$$\Gamma(x_k, x_{k-1}) = \Gamma(x_{k-1}) + \lambda(x_k, x_{k-1}), \tag{4}$$

where

$$\Gamma(x_k) = \min_{x_{k-1}} \Gamma(x_k, x_{k-1}) \tag{5}$$

and

$$\Gamma(x_0) = 0. \tag{6}$$

Each time $\Gamma(x_k)$ is obtained, the value of x_{k-1} associated with the minimization is stored, so that upon reaching the final state x_{N+1}, where $u_{N+1} = *$, it is possible to trace a complete path back from $k = N$ to $k = 1$. Under reasonable assumptions, it is possible to substitute for $-\ln P(z_k/u_k)$ the set of recognizer distances $d(u_k/z_k)$. The choice of ν determines the degree of letter-by-letter statistical dependence by which the directory is characterized. For example, for $\nu = 0$, $P(x_k/x_{k-1}) = P(u_k/u_{k-1})$ and, in effect, the directory is characterized by position-dependent diagrams. Note that for small values of ν it is possible to find as a solution a sequence which is not actually found in the directory. However, if ν is specified to be longer than the longest string in the directory, this outcome is excluded. In this case,

$$P(x_k/x_{k-1}) = P(x_k, x_{k-1})/P(x_{k-1}) = P(x_k)/P(x_{k-1}), \tag{7}$$

since, for ν large enough, x_k completely specifies x_{k-1}. $P(x_k)/P(x_{k-1})$ can be estimated by $N(x_k)/N(x_{k-1})$, where $N(x_k)$ is the number of occurrences of the sequence x_k in the directory. Substituting into eq. (4), we obtain

$$\Gamma(x_k, x_{k-1}) = \Gamma(x_{k-1}) + d(u_k/z_k) - \ln N(x_k)/N(x_{k-1}). \tag{8}$$

Note the important result that

$$\Gamma(x_k) = \Gamma(x_k, x_{k-1}) \tag{9}$$

since under the assumption for ν there is only one possible antecedent x_{k-1} to x_k.

A further modification which leads to an intuitively appealing result is obtained by replacing $-\ln N(x_k)/N(x_{k-1})$ by $\delta(x_k)$, where

$$\delta(x_k) = \begin{cases} 0 & \text{for} \quad N(x_k) > 0 \\ \infty & \text{for} \quad N(x_k) = 0. \end{cases} \tag{10}$$

The result of the iteration expressed by eq. (8) is then

$$x_N = \sum_{k=1}^{N} d(u_k/z_k) \tag{11}$$

for each x_N found in the directory. The solution then becomes that x_N for which $\Gamma(x_N)$, the accumulated sum of recognizer distances, is minimum.

This simplified procedure is illustrated by an example. A speaker has spelled the string "BERKLED", seeking a directory listing for D. Berkley. The last name has been truncated to six characters, and there is no separation between the letters of the last name and the initial. The recognizer distances $d(u_k/z_k)$ are shown in Table II. Figure 4 is a trellis diagram showing the search procedure for this string. Only those states are shown for which the transition distances $\lambda(x_k, x_{k-1})$ are finite. Over each transition arrow is the accumulated recognizer dis-

Table II—Example of letter candidates and corresponding recognizer distances for spelled input string "BERKLED"

Position	1	2	3	4	5	6	7
Spelled	B	E	R	K	L	E	D
	Candidate Letters and Distances						
	B 0.064	E 0.0183	I 0.160	K 0.101	L 0.114	E 0.059	P 0.089
	P 0.081	P 0.107	R 0.176	J 0.215		P 0.089	T 0.101
	E 0.091	B 0.112	O 0.189			B 0.090	D 0.111
	D 0.148					D 0.139	B 0.150
	G 0.154					G 0.144	Z 0.161

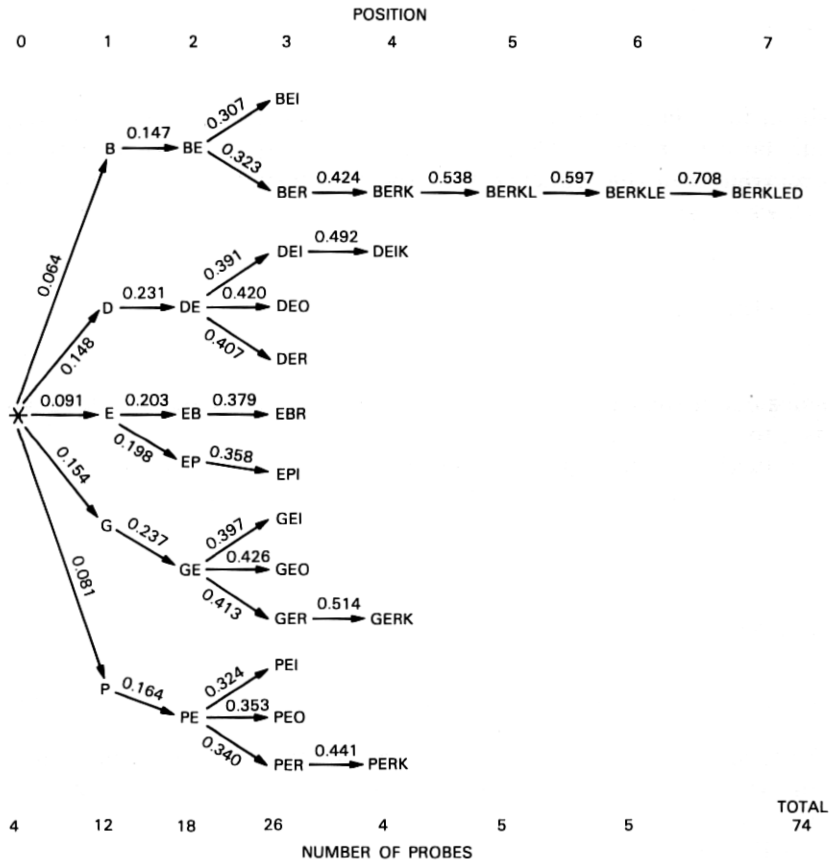


Fig. 4—Trellis diagram for the Viterbi search procedure corresponding to the example specified in Table II.

tance leading to the state on the right. In contrast to a trellis diagram for a general Viterbi algorithm procedure, there is just one incoming branch for each node. The only possible exception is the final accepting state, in which case, as already mentioned, the string with the least accumulated distance is taken as the solution.

For each u_k, there is a probe of the directory for every possible string prefix x_k to determine whether $N(x_k) > 0$, that is, whether one or more strings are listed in the directory with the prefix x_k. The number of probes for each k and the total is shown along the base of the figure for the example.

This procedure is exhaustive and has the optimum properties associated with Viterbi algorithm procedures. Nevertheless, it lacks some efficiencies. For example, suppose the best candidate letters provided by the recognizer form the actual intended string. With the Viterbi algorithm approach, it is necessary to probe the directory for each string prefix prescribed by the candidate letters before a decision can be made. Since the candidate string formed from the best candidate letters is the most likely match according to the recognizer, it seems natural to probe the directory directly and immediately for this string.

Potentially more efficient strategies than the Viterbi algorithm do exist for determining best paths through trellis-type structures. Two such procedures, which were first investigated in the problem area of decoding convolutional codes, are the Fano algorithm[10] and the Zigangirov-Jelinek stack algorithm.[11,12] These procedures are included in the general class of procedures known as "backtrack" techniques.[13-15] They have the common property that the better, more probable branches in the trellis are probed first in the search for the best complete path.

An empirical backtrack technique is the actual procedure used in this study for finding matching directory strings. Outlined in the flow chart in Fig. 5, it is an iterative procedure in which each trial is an attempt to match a complete candidate string to a directory string. If no match is obtained, a new candidate string is constructed by replacing the candidate letter in the first position of mismatch. The first position of mismatch is the leftmost mismatch position of the best matching directory string scanning from left to right. The substitution replaces the existing candidate letter with the next best candidate letter provided by the recognizer in that position. If there are no more candidate letters available in the mismatch position, the selected position is "backtracked" to the preceding position.

Thus two salient points are associated with this procedure. First, candidate letters are replaced in the order of merit provided by the recognizer under the assumption that the most likely string matches are to be affected by the better candidate letters. Second, the backtrack procedure provides a systematic and efficient strategy for constructing candidate strings and probing the directory. It has a property, common also to the Viterbi procedure, in that, once a string prefix is found to result in a mismatch, it never appears again as the prefix of subsequent candidate strings. The backtrack procedure is also potentially exhaustive. However, we choose to terminate the procedure prior to exhaustion once one or more matching directory strings have been found. The assumption here is that the first match is likely to be the correct match so that there is little justification to driving the procedure to exhaustion. An example illustrating the backtrack procedure is shown in Fig. 6 with the same recognizer candidates used in the previous example. What is shown is the sequence of candidate and corresponding best matching directory strings which occurs until a complete match is found. The mismatch position is indicated by a square drawn around the corresponding letter in the candidate string. Note that the

CANDIDATE LETTER ARRAY: $u_1(m_1), u_2(m_2), \cdots u_N(m_N)$

$m_k = 1, 2, \cdots M_k \quad 0 \le M_k \le M_{MAX}$

N = STRING LENGTH

CANDIDATE STRING: $v = (v_1, v_2, \cdots v_N)$

WHERE $v_k = u_k(m_k)$

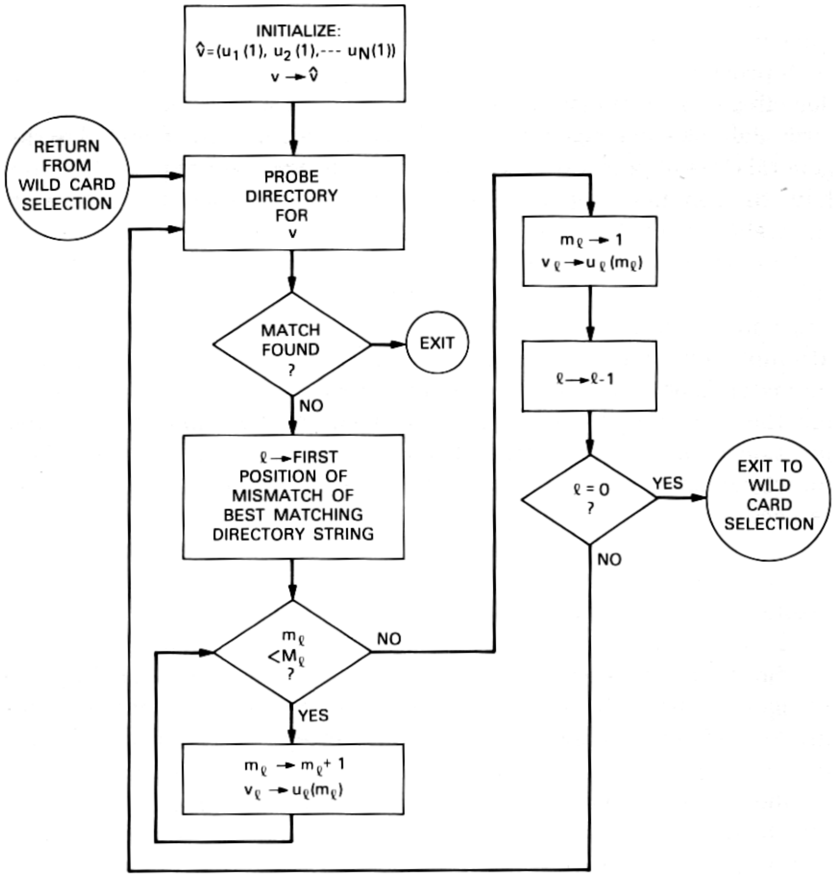

INITIALIZE:
$\hat{v} = (u_1(1), u_2(1), \cdots u_N(1))$
$v \to \hat{v}$
RETURN FROM WILD CARD SELECTION
PROBE DIRECTORY FOR v
MATCH FOUND ?
EXIT
NO
$\ell \to$ FIRST POSITION OF MISMATCH OF BEST MATCHING DIRECTORY STRING
$m_\ell < M_\ell$?
NO
YES
$m_\ell \to m_\ell + 1$
$v_\ell \to u_\ell(m_\ell)$
$m_\ell \to 1$
$v_\ell \to u_\ell(m_\ell)$
$\ell \to \ell - 1$
$\ell = 0$?
YES
EXIT TO WILD CARD SELECTION
NO

Fig. 5—Flow chart showing backtrack procedure for constructing candidate strings.

correct match is found after five probes of the directory compared with 74 probes required for the Viterbi procedure.

The efficiency and accuracy of this type of search is related to the ordering of the candidate letters in each position. The greater the number of correct letters found among the top candidates, the faster the search converges. In addition, since the search proceeds from left to right in the string, it proceeds faster if the correct letters are distributed among the high ranking candidates in the earlier positions rather than the later positions. In the example, if *B* were not the best

POSITION	1	2	3	4	5	6	7
SPELLED	B	E	R	K	L	E	D

CANDIDATE LETTERS

1	B	E	I	K	L	E	P
2	P	P	R	J		P	T
3	E	B	O			B	D
4	D					D	B
5	G					G	Z

CANDIDATE STRING NUMBER								
1	B	E	I	$\boxed{K}$	L	E	D	CANDIDATE STRING
	B	E	I	D	E	L	R	BEST MATCHING DIRECTORY STRING
2	B	E	I	$\boxed{J}$	L	E	P	
	B	E	I	D	E	L	R	
3	B	E	R	K	L	E	$\boxed{P}$	
	B	E	R	K	L	E	D	
4	B	E	R	K	L	E	$\boxed{T}$	
	B	E	R	K	L	E	D	
5	B	E	R	K	L	E	D	
	B	E	R	K	L	E	D	

Fig. 6—Backtrack search corresponding to the example specified in Table II.

candidate letter in position 1, many more probes might be required before a match is found.

On the average, 160 directory strings are accessed and compared for each candidate string. In the present implementation, this consumes approximately 1 s, or approximately 6.25 ms per comparison. In the evaluation to be described later, it has been found that the median number of candidate strings searched per trial is 3.5, corresponding to a total search time of 3.5 s.

As previously mentioned, it is possible for no candidate letters to be available in one or more positions in the string. In this instance, these positions are simply ignored, and a match is possible with any letter in these positions. Of course, the greater the number of letter positions which are ignored in a search, the greater the likelihood of an error. Somewhat arbitrarily, no more than three no-candidate positions are allowed in the present implementation. The directory range must include every listing which includes the letters A to Z in the no-candidate positions. Hence, no-candidate positions also cause an increase in the directory search range and consequently the search time. Since the directory is ordered alphabetically by string position from left to right, this effect is greater for no-candidate positions in the initial part of the string and critical when there are no candidates in the first position. Since the absence of a candidate in the first position would require searching the directory from beginning to end, which would not be practical, an alternate scheme has been devised. A secondary directory table is available which contains an alphabetically ordered list of listing fragments comprising positions 2, 3, and 4. For each entry in this table, a set of addresses locates listings in the main directory containing the specified listing fragment. Thus, whenever a candidate string with no candidate letters in the first position is presented, the secondary table is searched first for matching fragments which point to complete listings in the main directory. The resulting search time is negligibly greater than a search involving only the main directory.

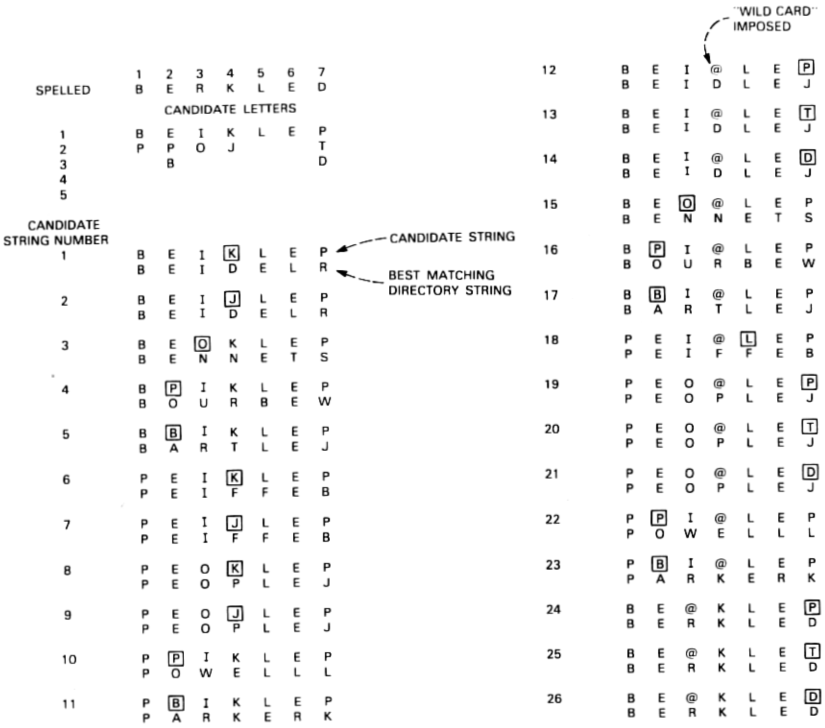
The flow chart in Fig. 2 indicates that the search fails with no matching directory listings found after all candidate strings have been exhausted. However, the "search fails" result does not occur after all candidate strings composed of candidate letters supplied by the acoustic recognizer have been exhausted. At this point, the composition of candidate strings is relaxed by *imposing* a "no candidate" or "ignore" condition on a selected position and the search is restarted. The reasoning behind this "wild card" procedure is as follows. Assuming that a directory listing actually exists corresponding to the customer's request, the exhaustion of candidate strings constructed from the array of candidate letters indicates that the candidates supplied by the recognizer in one or more positions are all incorrect. Suppose that the probability of the correct letter not appearing as a candidate in a given position is approximately 0.05.* A simple calculation shows that, given this probability, the probability of only incorrect candidates being found in more than one position of a string of length 8 is approximately 0.06.† Thus, for the most part, when a search fails to find a matching listing with candidate letters supplied by the recognizer, it would suffice to impose an "ignore" or "wild card" condition at just one position. The position in which the "wild card" is most likely to effect a match is the maximum (rightmost) mismatch position over all previous searches. The most likely candidate string is the one containing the prefix which generated this mismatch position.

The "wild card" procedure is illustrated by another example shown in Fig. 7. This example is a modification of Example 1 in which the correct candidate letter "R" is not present among the candidates in position 3. In addition, the number of candidate letters in positions 1, 6, and 7 has been reduced to simplify the example. At candidate string number 11 "PBIKLEP", all candidate strings comprising supplied candidate letters have been exhausted. The best mismatching strings among all the candidate strings are strings numbers 1, 2, 6, 7, 8, and 9, in all of which the mismatch position is 4. The "wild card" flow chart shown in Fig. 8 indicates that the initial "wild card" candidate string is constructed by imposing a "wild card" on the first best mismatching candidate string, which is string number 1 in the example. The resulting

* This value conforms with the results of the evaluation to be described.

† This figure results from evaluating:

$$P = 1 - [p^n + np^{n-1}(1-p)], \text{ where } 1 - p = 0.05 \text{ and } n = 8.$$



	1	2	3	4	5	6	7
SPELLED	B	E	R	K	L	E	D
CANDIDATE LETTERS							
1	B	E	I	K	L	E	P
2	P	P	O	J			T
3		B					D
4							
5							

CANDIDATE STRING NUMBER							
1	B	E	I	$\boxed{K}$	L	E	P
	B	E	I	D	E	L	R
2	B	E	I	$\boxed{J}$	L	E	P
	B	E	I	D	E	L	R
3	B	E	$\boxed{O}$	K	L	E	P
	B	E	N	N	E	T	S
4	B	$\boxed{P}$	I	K	L	E	P
	B	O	U	R	B	E	W
5	B	$\boxed{B}$	I	K	L	E	P
	B	A	R	T	L	E	J
6	P	E	I	$\boxed{K}$	L	E	P
	P	E	I	F	F	E	B
7	P	E	I	$\boxed{J}$	L	E	P
	P	E	I	F	F	E	B
8	P	E	O	$\boxed{K}$	L	E	P
	P	E	O	P	L	E	J
9	P	E	O	$\boxed{J}$	L	E	P
	P	E	O	P	L	E	J
10	P	$\boxed{P}$	I	K	L	E	P
	P	O	W	E	L	L	L
11	P	$\boxed{B}$	I	K	L	E	P
	P	A	R	K	E	R	K
12	B	E	I	@	L	E	$\boxed{P}$
	B	E	I	D	L	E	J
13	B	E	I	@	L	E	$\boxed{T}$
	B	E	I	D	L	E	J
14	B	E	I	@	L	E	$\boxed{D}$
	B	E	I	D	L	E	J
15	B	E	$\boxed{O}$	@	L	E	P
	B	E	N	N	E	T	S
16	B	$\boxed{P}$	I	@	L	E	P
	B	O	U	R	B	E	W
17	B	$\boxed{B}$	I	@	L	E	P
	B	A	R	T	L	E	J
18	P	E	I	@	$\boxed{L}$	E	P
	P	E	I	F	F	E	B
19	P	E	O	@	L	E	$\boxed{P}$
	P	E	O	P	L	E	J
20	P	E	O	@	L	E	$\boxed{T}$
	P	E	O	P	L	E	J
21	P	E	O	@	L	E	$\boxed{D}$
	P	E	O	P	L	E	J
22	P	$\boxed{P}$	I	@	L	E	P
	P	O	W	E	L	L	L
23	P	$\boxed{B}$	I	@	L	E	P
	P	A	R	K	E	R	K
24	B	E	@	K	L	E	$\boxed{P}$
	B	E	R	K	L	E	D
25	B	E	@	K	L	E	$\boxed{T}$
	B	E	R	K	L	E	D
26	B	E	@	K	L	E	$\boxed{D}$
	B	E	R	K	L	E	D

Fig. 7—Backtrack search example with imposed "wild card."

candidate string yields a mismatch in position 7 which is not resolved by the remaining candidate letters in that position (strings 13 and 14). The generation of candidate strings proceeds until all candidate strings are exhausted once more at candidate string number 23. The wild card is then backtracked to position 3, resulting in a match with candidate string number 26. A search results in a complete failure only when all candidate strings have been exhausted with the wild card backtracked to position 1.

III. EVALUATION

An evaluation was carried out to assess the performance of the system for a small sample of talkers. Ten adult talkers, six male and four female, participated in the experiment. Access to system was via an ordinary telephone handset over dialed-up lines through the local PBX. The system resides in a Data General Nova 840 laboratory computer interfaced to the telephone network through a Western Electric 407A data set. For each talker, two reference patterns were obtained for each of the 39 vocabulary items shown in Table I. These were obtained in two sessions of approximately 10 minutes each,

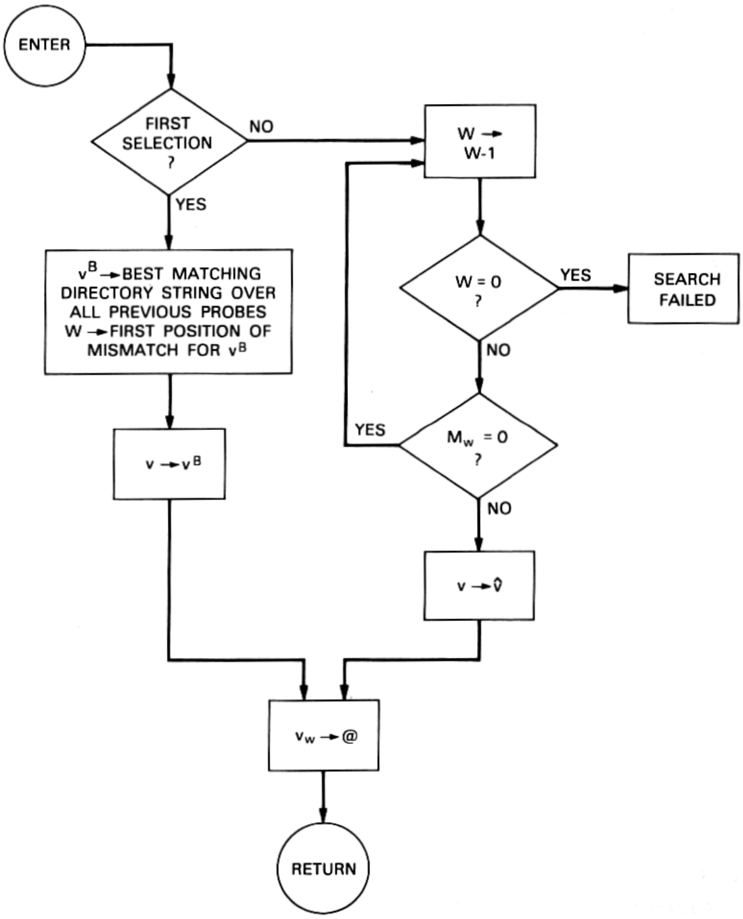

ENTER
FIRST SELECTION ?
NO
YES
W → W-1
v^B → BEST MATCHING DIRECTORY STRING OVER ALL PREVIOUS PROBES
W → FIRST POSITION OF MISMATCH FOR v^B
W = 0 ?
YES
SEARCH FAILED
NO
v → v^B
YES
M_W = 0 ?
NO
v → v̂
v_W → @
RETURN

Fig. 8—Flow chart showing "wild card" position selection procedure.

separated by at least one day. The talkers were seated at the computer display terminal and received visual and audible cues for recording utterances. Following the reference sessions, one or more test sessions took place for each talker in which the list of 50 names shown in Table III were spelled out. These names were randomly selected from the Bell Laboratories telephone directory. For spelled input, the last names are truncated to six letters. The remaining letters are enclosed in parentheses. There is a grand total of 364 spelled letters in the list, the distribution of which is shown in Fig. 9. Upon receipt of a cue, with the truncated spelling of the name displayed, a spelled name was recorded in the manner described in Section II. The acoustic analysis data for each spelled name were recorded on digital tape for subsequent off-line recognition and directory search. Two rejection threshold condi-

Table III—Test list of 50 names. For input, last names are truncated to 6 letters

1. ZBOYAN A M	26. TINLEY M A
2. KRIEGE(R) G E	27. SHAEFF(ER) P A
3. ROHM B J	28. LIND G R
4. EPWORTH(H) R	29. SHIPLE(Y) J W
5. LINDHA(RD) E A	30. CUCCO J A
6. BURNS J F	31. HOFER F R
7. RUDDOC(K) B	32. DUNBAR J J
8. SCHILL(O) R F	33. DUKE S D
9. GOOZH J L	34. WASSON R D
10. VIROST(EK) A M	35. HOOD A A
11. LENNON F W	36. MENGEL M R
12. VASHIS(HTA) P	37. RAVER D F
13. DUFFY G L	38. FULTZ K E
14. YAEGER J C	39. CADWEL(L) K
15. CRAWFO(RD) C D	40. YOUHAS J M
16. WEEKS C G	41. VANBEN(THEM) J
17. AVEYAR(D) R L	42. DUNCAN(SON) J P
18. GRECO T J	43. SUMNER EE
19. MODARR(ESSI) A R	44. LAWREN(Z) D A
20. ERWIN W J	45. BLY J
21. LUM P S	46. NEWELL J A
22. SOOS N A	47. STAUBA(CH) W E
23. SKARIN R H	48. TATE B A
24. TENEYC(K) J H	49. ONDER J J
25. SACCO G A	50. SOLOMI(TA) K S

tions were observed for the recognition phase. The effect of relaxing the rejection threshold is to admit more candidates, which in turn increases the number of candidate strings and the time required for the directory search. For three of the 10 talkers, performance improved under a more relaxed threshold condition. For simplicity, the results are presented combining the two threshold conditions by including the results from the threshold condition that was best for each talker.

Word error rate, that is, acoustic error rate per spoken letter, is shown plotted as a function of number of best candidates in Fig. 10. The median across the 10 talkers is shown bracketed by the greatest and smallest individual error rates. This figure shows the rates for which the correct letter is not the best candidate, the best two candidates, etc. Thus the median rate at which the correct letter is not the best candidate is 20.5 percent, while the median rate at which the correct letter is not among the best five candidates is 4 percent. Improvement in performance diminishes rapidly as a function of the number of best candidates. Thus, there is likely to be only a negligible gain in performance if more than five best candidates are admitted.

Individual string error rates are shown as a function of individual word error rates—five best candidates in Fig. 11. String error rate is defined as the sum of string mismatch and no match rates. A mismatch is defined as obtaining an incorrect entry from the directory while no match is defined as finding no entry at all, a search failure. The median string error rate over all talkers, indicated by the arrow in the figure,

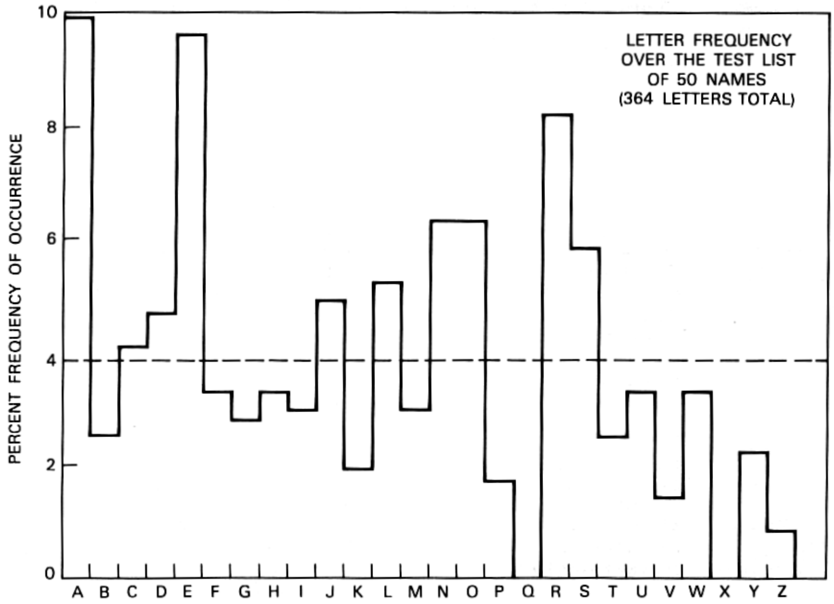


Fig. 9—Distribution of letters over the test list of 50 names.

is 4 percent, representing two errors over the 50-name list. The average mismatch rate is approximately four times greater than the average no-match rate.

A regression line has been fit among the individual points for string error rate plotted against word error rate (five best candidates). The coefficient of fit* is 0.78 indicating that word error rate (five best candidates) is a good predictor for string error rate. (This result is similar to the one obtained in Levinson et al.[16]) Attempts to fit a regression line among individual points of string error rate versus word error rate (less than five candidates) were not successful. This result is natural, since a correct directory match is highly dependent on the presence of the correct letter for each position among all the candidates presented for the search.

An important attendant result of this experiment is a tabulation of confusions made by the recognizer among the spoken letters. A confusion matrix is shown in Table IV. It provides the frequency of recognition for a given specified letter over all the trials of all the talkers. For the purpose of this tabulation, a letter is said to be recognized if it is ranked equal to or higher than the specified letter. In other words, an incorrect letter is counted each time it is a better candidate than the specified letter, while the specified letter is counted

* Estimate of the square of the correlation coefficient.

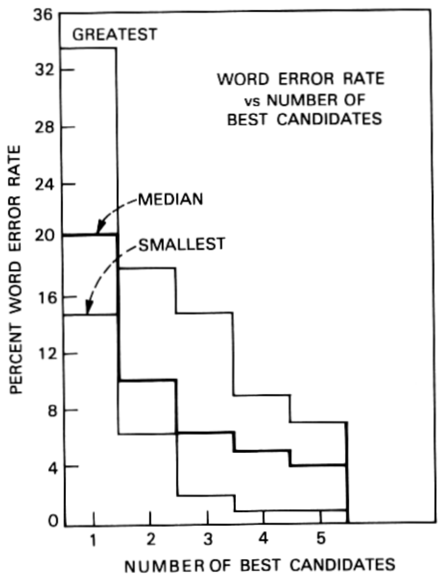


Fig. 10—Word error rate as a function of number of best candidates showing the median over the talker set as well as the greatest and smallest individual talker rates.

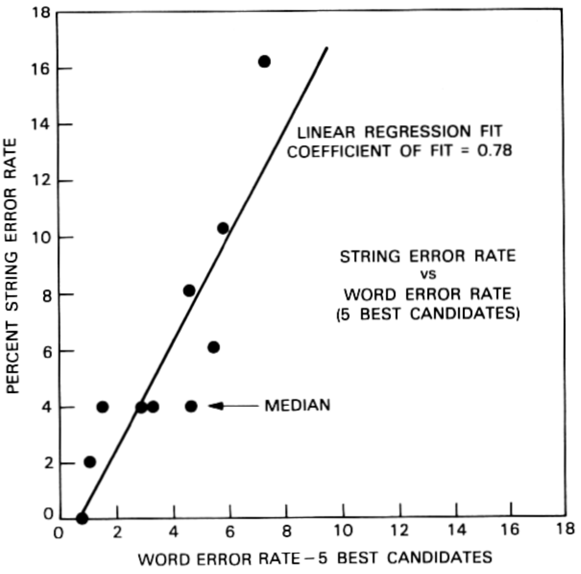


Fig. 11—Individual string error rates as a function of word error rate—5 best candidates.

Table IV—Confusion matrix showing frequency of recognition as a function of specified letter

	Percent Recognized as																	
Specified as	B	D	E	G	P	T	V	Z	C	A	J	K	I	Y	F	S	M	N
B (2.47)	36	21	9	0	9	5	11	0	0									
D (4.67)	9	42	7	7	12	8	10	0	0									
E (9.62)	9	7	44	0	16	11	7	0	0									
G (2.75)	0	5	0	65	0	8	0	5	0									
P (1.65)	6	6	0	0	56	14	9	5	0									
T (2.47)	0	9	0	5	16	55	6	0	5									
V (1.37)	7	10	0	8	9	6	49	0	5									
Z (0.82)	0	5	0	0	7	5	23	45	10									
C (4.12)	0	0	0	5	5	7	11	16	52									
A (9.89)									0	74	7	7						
J (4.94)									0	0	85	13						
K (1.92)									7	0	9	73						
I (3.02)													80	12				
Y (2.20)													0	96				
F (3.30)															90	10		
S (5.77)															8	89		
M (3.02)																	80	13
N (6.32)																	6	89

each time it appears in the list of candidates. Confusions less than 5 percent are ignored to simplify the matrix. The frequency of specification, from Fig. 9, is given in parentheses with each specified letter. There are 18 letters for which significant confusions are found. These letters are arranged into five groups with almost no interaction between the groups. The largest group, as expected, is the "BDE ···" family. Note that the confusions are rarely symmetric. For example, "B" is recognized as "D" 21 percent of the time, but "D" is recognized as "B" only 9 percent; "Z" is recognized as "V" 23 percent, but there is no significant confusion vice versa.

A different viewpoint is obtained by tabulating frequency of specification as a function of recognized letter. This tabulation, shown in Table V, specifically includes the *a priori* distribution of letters among the spelled names in the list. The relation between this frequency and the frequency of recognition as a function of specification is entirely analogous to the relation between *a posteriori* probability and channel probability in the specification of an information channel in information theory. Let $f(j_r/i_s)$ be the frequency of recognition of letter j_r given the specification of letter i_s. This frequency is analogous to channel probability and is an intrinsic property of the recognizer. According to Bayes' Law, $f(i_s/j_r)$, the frequency of specification of letter i_s given recognition of letter j_r (analogous to *a posteriori* probability), is given by

$$f(i_s/j_r) = \frac{f(j_r/i_s)f(i_s)}{\sum\limits_{i_s} f(j_r/i_s)f(i_s)},$$

where $f(i_s)$ is the *a priori* frequency of letter i_s, and the summation in the denominator is carried out over all specified letters. Only for the special case when $f(i_s)$ is uniformly distributed, that, is equal to a constant, is $f(i_s/j_r)$ a function only of recognizer probabilities. It can then be appreciated that, since in the present situation the *a priori* distribution of letters is quite nonuniform, the effect on the *a posteriori* frequency of specification is highly significant. For example, when "P" is recognized, the specified letter is twice as likely to be an "E" than a "P." This makes some sense when it is realized that there are approximately six times as many E's in the list as P's.

The efficiency of the system is a function of the number of candidate strings searched until a match is obtained. The median number of candidate strings searched per spelled name for each individual talker is shown plotted as a function of string error rate. The median over the individual medians is 3.5. There is a fair correlation between number of strings searched and string error rate.

The percentage of trials for each talker which ended up as "wild card" searches is also shown plotted as a function of string-error rate

Table V—Confusion matrix showing frequency of specification as a function of recognized letter

	Percent Specified as																	
	(2.47)	(4.67)	(9.66)	(2.75)	(1.65)	(2.47)	(1.37)	(0.82)	(4.12)	(9.89)	(4.94)	(1.92)	(3.02)	(2.20)	(3.30)	(5.77)	(3.02)	(6.32)
Recognized as	B	D	E	G	P	T	V	Z	C	A	J	K	I	Y	F	S	M	N
B	39	17	32	0	0	0	0	0	0									
D	15	50	16	0	0	5	0	0	0									
E	5	7	83	0	0	0	0	0	0									
G	5	14	10	49	0	0	0	0	8									
P	5	14	38	0	19	0	0	0	0									
T	0	11	29	0	5	33	0	0	7									
V	11	16	22	0	0	0	18	6	13									
Z	0	10	10	7	0	0	0	22	35									
C	0	0	0	0	0	0	0	0	75									
A										98	0	0						
J										15	81	0						
K										26	19	50						
I													96	0				
Y													17	82				
F															85	13		
S															6	93		
M																	86	12
N																	7	92

for individual talkers in Fig. 12. The rate of "wild card" searches also seems to correlate fairly well with string error rate. The median number of probes for a "wild card" search is approximately 50, more than 10 times the median for all searches. The string error rate for "wild card" searches is also very much higher than the overall string error rate, the median being approximately 19 percent.

An important fact which bears on the number of "wild card" searches and the "wild card" error rate is that a search cannot succeed if more than one position in the string has only incorrect candidates. There were nine such trials in the evaluation, contributing to nearly 50 percent of the "wild card" error rate.

IV. DISCUSSION

The most important feature of this system, borne out by the evaluation, is the powerful correcting influence of the context of spelled names as listed in a telephone directory on the recognition of spoken spelled letters. Given a median word error rate of approximately 20 percent on spoken spelled letters, the median string error rate is 4 percent. This result is entirely similar to the result observed by Levinson et al.[16] in which syntax constraints were seen to have a powerful correcting influence on the same automatic word recognizer

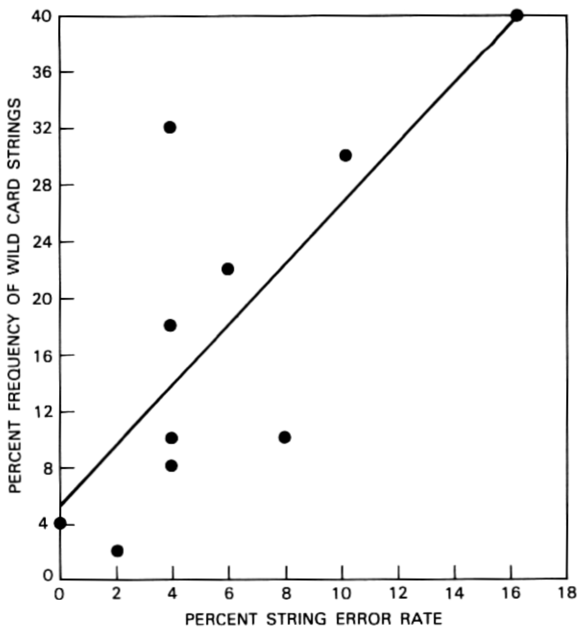


Fig. 12—Individual frequencies of "wild card" string searches as a function of string error rate.

for a 127-word flight information vocabulary. The error rate for parsed sentences was approximately 4 percent, while the acoustic word error was approximately 11 percent.

It has been noted previously that the vocabulary of spelled letters is a particularly difficult vocabulary for an automatic word recognizer due to the large number of minimal distinctions between words. This may account for the large difference in word error rates between the airline and spelled letter vocabularies. It should be noted that Itakura[1] obtained an 11.4 percent word error rate for the vocabulary of spelled letters plus digits. This is lower than the lowest individual error rate obtained in the present experiment (Fig. 10). However, the results are not directly comparable, since digits are much easier to recognize than spelled letters.

It is of interest to examine in some detail the kinds of errors obtained in the evaluation. First of all, recall from the discussion in Section II that if only incorrect candidate letters are found in more than one position the search is guaranteed to fail. This is because the "wild card" procedure imposes an "ignore" condition on just one letter position at a time. It was noted that for strings of length 8 and a word error rate (five best candidates) of 5 percent, the estimated rate at which only incorrect letters are found in more than one position is 6 percent. This, then, is the estimated rate of guaranteed failures. The situation deteriorates rapidly with increasing word error rate. If word error rate (five best candidates) increases to 8 percent, the rate of guaranteed failures can be expected to increase to 13 percent. The two talkers with the worst word error rate (five best candidates) performances of 7.4 and 5.9 percent generate guaranteed failures at rates of 6 and 8 percent, respectively. These figures, while not as bad as the predicted 13 percent for an 8-percent word error rate, represent a serious weakness in the system. There were just two guaranteed failures among all the remaining talkers.

After excluding errors associated with imposed "wild cards," the remaining string errors occur because of the existence in the directory of incorrect strings whose letters match candidate letters ranking higher than those corresponding to the correct string. Out of all the trials over all the talkers in the evaluation, there were six trials, or approximately 1 percent of the total, for which the above condition was true and for which errors were obtained. One particularly persistent example, accounting for four of the six errors, involved the test string "VANBEN*J*'. In each case, in the fourth position "D" was a better candidate than the specified letter "B," resulting in the incorrect matching string "VANDEN*J*". This situation is associated with the conjunction of two adverse conditions. These are acoustic letter confusions of high probability and directory entries with highly similar name strings.

A similar situation which might have been expected to produce more mismatches than it did in the evaluation is mismatching initials even though last-name strings are correctly matched.* A string of initials of length 1 or 2 is a far weaker constraint on acoustic errors than the last-name string, which is generally length 6. For the most common last names, where there are many entries differing only in initials, such mismatches may be serious. (As presently implemented, this condition can actually produce worse results since matches are allowed even if second initials are not supplied by the talker or do not match the second initial in the listing string.)

A possible way to resolve the kinds of mismatches described here is to request the talker to specify additional information.† For example, if an additional last name character is requested for the example cited, there is little chance of mismatch between "VANBENT*J*" and "VANDENB*J*".

Although there were 88 "wild card" searches out of 500, there were only 34 searches in which one or more string positions had no candidates at all. Of these, only one string had more than one such position, and only one string generated an error. This low incidence of no-candidate positions is a consequence of the relaxed threshold conditions that were imposed. Preliminary experimentation had indicated a preference for admitting more rather than less candidates, with a greater likelihood for the correct letter to be among them.

Finally, inspection of Table V suggests a possible revision to the system. The table indicates which letters are likely to have been specified when a given letter is recognized, and accordingly accounts for both the *a priori* distribution of letters and the characteristics of the acoustic recognizer. Improved performance might be obtained if letters are included as candidates when the table indicates they are likely candidates, whether or not they are included as candidates by the recognizer on a given trial. With respect to the example cited in the results, it seems reasonable always to include "E" as a candidate when "P" is recognized, since the table indicates that "E" is twice as likely to have been the specified candidate.

Other possible modifications to the system come to mind. As has been emphasized, the present system is guaranteed to fail if there are only incorrect candidates in more than one position. The "wild card" process imposing the "wild card" position one at a time is the limiting factor. An additional "wild card" process can be designed which allows for two such positions. However, the process would likely be cumbersome and protracted. Moreover, it could well be self-defeating by the

* Only four mismatches were obtained.

† As mentioned previously, listing ambiguities are resolved by requesting the talker to specify location codes or organization numbers.

admission of more false or unintended matches rather than achieving better compensation for acoustical errors.

A worthwhile endeavor which would help resolve many of the uncertainties encountered in this study would be to determine quantitatively what statistical characteristics both the directory and the recognizer should have to attain a specified string error rate.

V. CONCLUSION

A means for obtaining voice-actuated directory assistance has been proposed, implemented, and evaluated. Requests made in the form of spoken spelled names can be executed reasonably reliably despite imperfect acoustic recognition. This is accomplished by offering alternate spellings of the name string to be searched in the directory, relying on the spelled context of the correct name to restrict matches to the entries that have been requested. Reliability as well as efficiency also results from the rank-ordering of possible spellings as prescribed by the recognizer combined with the systematic (alphabetic) organization of the directory. Such a system, of course, is not restricted to furnishing directory information but may be applied to any dictionary-like store in which the spelled context provides reasonable constraints.

The 4-percent string error rate obtained in the evaluation, while most encouraging in comparison with the 20-percent acoustic word error rate, does not seem adequate for actual directory assistance service. An immediate goal is improvement of the acoustic recognizer to provide word error rates in the range of 10 percent to provide, in turn, string error rates in the order of 1 to 2 percent.

VI. ACKNOWLEDGMENT

The authors wish to express their appreciation to M. E. Lesk for providing the *TOUCH-TONE* system directory access program and to J. F. Gimpel and N. F. Maxemchuk for helpful discussions on backtrack procedures and decoding techniques.

REFERENCES

1. M. E. Lesk and C. A. McGonegal, "User-Operated Directory Assistance," unpublished work.
2. L. R. Rabiner and R. W. Schafer, "Digital Techniques for Computer Voice Response: Implementations and Applications," Proc. IEEE, *64* (1976), pp. 401–576.
3. L. H. Rosenthal, et al., "A Multiline Computer Voice Response System Using ADPCM Coded Speech," IEEE Trans. Acoustics, Speech and Signal Proc., *ASSP-22* (1974), pp. 339–352.
4. F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. Acoustics, Speech and Signal Proc., *ASSP-23* (1975), pp. 67–72.
5. A. E. Rosenberg and F. Itakura, "Evaluation of an Automatic Word Recognition System over Dialed-Up Telephone Lines," J. Acoust. Soc. Am., *60,* Suppl. No. 1 (1976), p. 512.
6. D. E. Knuth, "The Art of Computer Programming," Vol. 3, *Searching and Sorting*, New York: Addison-Wesley, 1973.

7. G. D. Forney, "The Viterbi Algorithm, Proc. IEEE, *61* (1973), pp. 268–278.
8. D. L. Neuhoff, "The Viterbi Algorithm as an Aid in Text Recognition," IEEE Trans. Inform. Theory, *IT-21* (1975), pp. 222–226.
9. C. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. Inform. Theory, *IT-13* (1967), pp. 260–269.
10. R. M. Fano, "A Heuristic Discussion of Probabilistic Decoding," IEEE Trans. Inform Theory, *IT-9* (1963), pp. 64–74.
11. K. Sh. Zigangirov, "Some Sequential Decoding Procedures," Problemi Peredachi Informatsii, *2* (1966), pp. 13–25.
12. F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack," IBM J. Research and Development, *13* (1969), pp. 675–685.
13. S. W. Golomb and L. D. Baumert, "Backtrack Programming," J. ACM, *12* (1965), pp. 516–524.
14. A. Nijenhuis and H. S. Wilf, *Combinatorial Algorithms,* New York: Academic Press, 1978.
15. E. G. Whitehead, "Combinatorial Algorithms," Lecture notes published by Courant Institute of Mathematics, New York University, 1973.
16. S. E. Levinson, A. E. Rosenberg and J. L. Flanagan, "Evaluation of a Word Recognition System using Syntax Analysis," B.S.T.J., *57,* No. 5 (May–June 1978), pp. 1619–1626.