THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 59, No. 8, October 1980
Printed in U.S.A.

A Minimum-Distance Search Technique and its Application to Automatic Directory Assistance

By B. ALDEFELD, S. E. LEVINSON, and T. G. SZYMANSKI

(Manuscript received August 31, 1979)

This paper describes a new search procedure and its application to the problem of obtaining telephone directory information from spoken spelled input. The method obtains its speed from using the concept of equivalence classes, with names classified according to their letter-by-letter acoustic similarity. It derives its accuracy from the use of a minimum-distance criterion for selecting answers. The search procedure finds the name with the minimum distance, usually after only a small fraction of the directory file has been examined. Using an acoustic analyzer with an 80 percent correct recognition rate for individual letters, a 98.6 percent correct recognition rate for names was achieved when the method was applied to a directory of 18,000 entries. On the average, only 1.2 percent of the directory had to be examined for each query. With an input recognition rate of 71 percent for letters, the respective figures were 97.2 percent and 2.8 percent.

I. INTRODUCTION

Automatic methods for obtaining telephone directory information from spoken spelled input can be highly useful. This directory assistance task, however, is one of considerable difficulty in which the problems of automatic speech recognition are exacerbated by the confusability of the vocabulary involved. This is especially true when telephone quality speech is used, since the acoustic differences within certain subsets of the alphabet are extremely small. In particular, such similar-sounding letters as "A," "J," and "K," or "B," "D," "P," and "V" are likely to be confused.

In this paper, we consider systems in which the recognition of names takes place in two distinct stages, an *acoustic analysis* phase, in which the individual letters of the input are recognized, and a *search* phase, during which a directory is interrogated. We assume that the acoustic

analyzer and its performance are given and focus our attention on devising a search method that allows the correct recognition of spelled names despite the errors introduced during the recognition of the input letters.

Recently, Rosenberg and Schmidt[1] have described a method which achieves a 92-percent correct recognition rate for names when driven by an acoustic analyzer with an 80-percent correct recognition rate for individual letters; their results were obtained using an 18,000-entry directory. Their method is to look in the directory for each name in a sequence of candidate names constructed from the output of the acoustic analyzer. The search continues until a match is found.

Since the method employed by Rosenberg and Schmidt ignores the distances provided by the acoustic analyzer for each letter, making use only of letters' rank ordering, it should be possible to improve the accuracy of the search procedure. Moreover, since each interrogation of the directory entails an access of a secondary storage file, a relatively slow operation, it can be hoped that the speed of the method can be increased.

The basic idea underlying the search procedure described in this paper is the use of equivalence classes into which all strings are grouped according to their letter-by-letter acoustic similarity. We have developed a search technique based on this classification that usually searches only a few small, contiguous portions of the directory file before the most closely matching name is found. The use of a minimum-distance criterion for selecting names provides high recognition accuracy.

In the next section, the directory assistance problem is described in more detail. Section III then gives a general description of the classification and search technique. A particular implementation and some experiments with it are described in Section IV. Section V concludes the paper with a brief summary of the method and results.

II THE DIRECTORY ASSISTANCE PROBLEM

Assume that we are given a directory with a possibly large number of entries, where an entry consists of a person's name together with some associated information. For the purpose of this paper, a name is considered to be a single string of letters, comprising a surname and initials. The problem to be solved is determining that entry which, in some sense, most satisfactorily matches the spoken spelled input. Making the associated information available is then a trivial task.

We assume that when the acoustic analyzer receives a spoken letter, it compares the properties of this utterance with a set of reference patterns for the letters of the alphabet. The output is given as a column of real numbers, or *distances*, which are a measure of dissim-

Table I—Sample D and L matrices for the spoken string "WHITE"

0.38	0.39	0.39	0.27	0.28	(W)	(H)	(I)	D	(E)
0.65	0.43	0.40	0.28	0.29	N	X	Y	P	D
0.66	0.59	0.51	0.30	0.30	Q	S	R	G	B
0.75	0.66	0.55	0.34	0.31	O	A	M	(T)	P
0.75	0.68	0.61	0.34	0.31	U	F	W	C	G

ilarity between the input and reference patterns, a low value indicating a high similarity. For the entire spoken string with n letters, the information provided by the recognizer is conveniently represented by a matrix D of distances and L of corresponding letters,

$$D = [\delta_{i,j}] \qquad 1 \le i \le 26 \qquad 1 \le j \le n \tag{1}$$

$$L = [\lambda_{i,j}] \qquad 1 \le i \le 26 \qquad 1 \le j \le n, \tag{2}$$

where each column corresponds to one of the spoken input letters. Specifically, $\delta_{i,j}$ is the distance from the pattern of the jth spoken utterance to the reference pattern of letter $\lambda_{i,j}$. Conveniently, the distances within each column of D are ordered by increasing values. Table I shows an example of such matrices, truncated to five rows, obtained from spelling the string "WHITE."

Because of the uncertainty introduced by the acoustic analyzer, it is usually not obvious how to recover the spoken name from the matrices. Using an analyzer with a correct recognition rate for letters of x, the probability of finding a spoken name of n letters in the first row of the L matrix is only x^n, which, for example, evaluates to 0.17 for $x = 0.8$ and $n = 8$. In general, one or more of the letters in the spoken name will have relatively high ranking positions in the L matrix (i.e., will have relatively large distances in the D matrix).

A minimum-distance criterion is an obvious choice for a decision rule for the selection of a name from the directory. Under a suitable extension of the distance metric from individual letters to strings, the name with the minimum total distance should have a very high probability of being correct. We stipulate in this paper that a suitable total distance is given by the sum of the individual distances. This is the same stipulation successfully employed in the widely used Itakura distance measure.[2]

We can now formalize the directory assistance problem in the following terms: Given the information contained in the D and L matrices, find that name in the directory that has the minimum distance, with the distance of a string being defined as

$$D(\lambda_{i_1,1}\lambda_{i_2,2}\cdots\lambda_{i_n,n}) = \sum_{1\le j\le n} \delta_{i_j,j}. \tag{3}$$

If two or more names have the minimum distance, we require that they all be found.

Two obvious methods exist to solve this search problem, both of which, however, are impractical. With the first of these, the desired entry is found by evaluating the distances of all names in the directory and selecting the name having the smallest distance. Of course, even for moderate-size directories, this exhaustive search technique is out of the question. The second approach is based on the enumeration of all n-letter strings in order of increasing distances, taking the first string that matches a name in the directory as the solution. A closer investigation shows that this method, too, is excessively time-consuming, primarily due to the large number of candidate strings generated before one is found which is in the directory. A test implementation, run under the conditions described in Section IV, required the generation of more than 1000 candidate strings in over 12 percent of the queries tried. Since each test of a candidate string for membership in the directory requires an access of secondary storage, this procedure is clearly too slow.

III. GENERAL DESCRIPTION OF THE SEARCH PROCEDURE

Our approach is based on the fact that a confusion of letters by the acoustic analyzer occurs primarily only within certain classes of acoustically similar letters. Thus, while a spoken letter may often be recognized incorrectly, the class to which it belongs is recognized with a high probability. This is used to advantage in our approach in that we choose the organization of the directory file and the search strategy to exploit this behavior of the acoustic analyzer.

3.1 Classification

The underlying idea is that of imposing an equivalence relation, $\equiv$, on the alphabet, by which all letters are classified according to their acoustic similarity. This equivalence relation is extended to strings of letters $l_1 l_2 \cdots l_n$ by defining $l_1 l_2 \cdots l_n \equiv l'_1 l'_2 \cdots l'_n$ if and only if $l_j \equiv l'_j$ for $1 \leq j \leq n$. In order to distinguish between equivalence classes of single letters and equivalence classes of strings, we shall refer to "letter classes" and "string classes" respectively. The number of different string classes, N_L, for strings of length n is related to the number of letter classes, N_l, by the simple equation $N_L = N_l^n$.

For the purpose of identifying string classes, it is convenient to assign a unique integer to each of them. One simple method for doing this is to assign each letter class a unique integer in the range 0 through $N_l - 1$. We can then calculate the identification number of a string class containing the string $l_1 l_2 \cdots l_n$ according to

$$M = \sum_{1 \leq j \leq n} m_j N_l^{n-j}, \tag{4}$$

where the m_j are the letter-class identifications corresponding to the l_j. In other words, we can find a string's class number by replacing each letter by the number of its letter class and interpreting the result as a number represented in base N_l.

For the purpose of organizing the directory file according to equivalence classes, we consider all names in the directory to consist of the same number of letters. This is not, of course, a restriction on the method, since a dummy letter, "blank," can be appended to the names as required. The reorganization of the directory is now easily performed by determining the string class of each name and storing the corresponding entry in the space reserved for that class. The set of directory names corresponding to a given equivalence class will be called a directory *block*.

Based on the classification of letters and strings, we introduce the notion of string-class distances. Let $h\ (l_1 l_2 \cdots l_n)$ represent the equivalence class of $l_1 l_2 \cdots l_n$, that is, the set of strings that are equivalent to $l_1 l_2 \cdots l_n$. Then the distance of the class $h\ (l_2 l_2 \cdots l_n)$ is defined by

$$D(h(l_1 l_2 \cdots l_n)) = \min\{D(l_1' l_2' \cdots l_n') \mid l_1' l_2' \cdots l_n' \equiv l_1 l_2 \cdots l_n\}. \quad (5)$$

Thus a class distance is the minimum distance of any members in that class. Class distances are easily computed by means of the identity

$$D(h(l_1 \cdots l_n)) = \sum_{1 \le j \le n} \min\{\delta_{i,j} \mid \lambda_{i,j} \equiv l_j\}. \quad (6)$$

Since not all strings in a given string class are necessarily in the directory, the string-class distance of any block of the directory may be less than the distance of any string in that block. The importance of the string-class distances is that they can be used to effectively guide the search through the blocks of the directory, as is explained below.

3.2 The search algorithm

Based on the preceding directory structure, we can formulate the search algorithm as follows.

(*i*) From the D and L matrices, as provided by the acoustic analyzer, calculate the distances of all string classes.

(*ii*) Select that string class with the smallest distance among the classes not yet examined. If two or more classes have this distance, select one arbitrarily.

(*iii*) Access the corresponding directory block. Calculate the distances of all names within that block and find the minimum.

(*iv*) Keep all entries corresponding to this minimum as solution candidates if this value is less than or equal to any distance found in the previously examined blocks.

(*v*) Check whether or not any string class not yet examined has a

distance less than or equal to the minimum distance found so far. If such a class exists, continue with step (*ii*), otherwise, terminate the procedure.

Two points deserve emphasis, as they determine the efficiency of this algorithm. The first concerns the order in which the blocks of the directory are examined. Choosing this order according to increasing string-class distance usually allows the desired entry to be found without examining the whole directory. Usually, only a small portion of the directory needs to be examined if the alphabet is suitably partitioned. A good choice is a partition in which confusable letters are always in the same class. In the ideal case when no confusions occur between classes, the desired entry is always located in the first class examined.

The second point concerns the stopping criterion of step (*v*). It ensures that, at termination of the procedure, no name in the directory can have a lower distance than that of the kept candidates. Thus, this algorithm guarantees that the names with the minimum distance will always be found. If only one such candidate is found, the query is uniquely answered. If two or more candidates are found, further information can be requested from the user to resolve the ambiguity.

A particular advantage of the method concerns the practical aspect of computer implementation. A large directory will in general be stored on secondary storage, for instance on a disk, and each access to this storage requires a considerable amount of time. In fact, the total search time will probably be dominated by these accesses. However, with a directory organization according to equivalence classes and a suitable choice of these classes, the number of accesses to secondary storage can be greatly reduced. In the ideal case mentioned above, only a single access is necessary.

The time needed to access secondary storage is one reason why the present method is superior to the method of straightforward enumeration of candidate names in order of increasing distance. Consider, for example, the same matrices of Table I. Construction of candidates in order of increasing distances, beginning with "WHIDE," "WHYDE," "WHIPE," etc., obviously requires many fruitless accesses to the directory until the correct string, "WHITE," is found. On the other hand, assuming an equivalence relation in which "B" ≡ "C" ≡ "D" = "E" ≡ "G" ≡ "P" ≡ "T," the string "WHITE" belongs to the same string class as "WHIDE," which is the string with the lowest distance. Thus, the correct name is encountered in the first directory block examined. In analogy to the simple method of enumerating all strings, our method can be viewed as enumerating all string classes in order of increasing string-class distance. The directory contents of each string class are examined when the class is enumerated, with each such examination requiring one access to secondary storage.

A number of modifications to the algorithm are possible that might be advantageous in a practical system. First, it might be desirable to include in the list of candidates not only all minimum-distance entries, but also those entries whose distances are within some small amount Δ of the minimum. If the distances of the candidates indicate a high probability of confusion, the system can request a repeat of the query or additional information to resolve the uncertainty.

Another modification of the procedure regards the determination of the sequence in which the string classes are examined. Instead of calculating the entire set of string-class distances at the beginning of the search, it is also possible to determine the next string class to be searched incrementally from the D and L matrices. This alternative can be preferable if a very large number of classes is used. An efficient algorithm to perform this task is described in Ref. 3.

A further modification of the procedure might be advantageous under certain circumstances. If the number of members in a string class is small compared to the limit given by the available space in primary memory, it can be more economical to read several classes in a single access. All these classes can then be examined together without noticeable increase in elapsed time. The procedure of directory organization and search remains essentially the same, except that string classes are grouped together into *buckets*. The distance of a bucket can be defined as the minimum distance of the classes it contains. So long as the mapping from classes to buckets is easily invertible, that is, so long as it is easy to determine what classes reside in a given bucket, it is easy to compute the distances of individual buckets and examine them in ascending order. Alternatively, it is possible to enumerate the string classes incrementally and examine the entire bucket containing each class, effectively simultaneously examining several classes.

Finally, we note that the method described also lends itself to some types of partial match retrieval problems. If the values in one or several columns of the distance matrix are not specified, equal values can be assigned to those columns with "don't care" letters. The search procedure can be applied in exactly the way described above, but a larger number of classes will have to be examined. Similarly, an input letter that is so hopelessly garbled that it is useless can still be handled with this technique.

IV. AN APPLICATION

The system to which we have applied the procedure described above uses a maximum number of eight letters in the spoken input, with six letters allowed for the last name and two for the initials. In addition to the letters of the alphabet, the spoken command "stop" is used to separate the last name from the initials and to designate the end of the input speech stream. Since this command is nearly always recog-

Table II—Performance of the acoustic analyzer

Rank i in Distance Matrix	Probability for Spoken Letter to Occur in Rank i		Probability for Spoken Letter to Occur in First i Ranks	
	Speaker Independent	Speaker Dependent	Speaker Independent	Speaker Dependent
1	0.71	0.80	0.71	0.80
2	0.14	0.096	0.85	0.896
3	0.062	0.047	0.912	0.943
4	0.026	0.020	0.938	0.963
5	0.020	0.012	0.958	0.975
6	0.013	0.010	0.971	0.985
7	0.008	0.006	0.979	0.991

nized correctly and since we do not include it in the classification, it need not be considered in the present context. We assume that the recognizer produces exactly one column of the D and L matrices for each spoken letter; that is, no letters are dropped, nor are any specious letters inserted.

Details of the acoustic analyzer used in our experiments are described in Refs. 1 and 4. The performance of the analyzer in recognizing the letters of the alphabet is shown in Table II. The data, taken from the study reported in Ref. 4, represent average values obtained over approximately 3600 letters using both speaker-dependent and speaker-independent reference patterns in the acoustic analyzer. The directory used in our tests was the same as that used by Rosenberg and Schmidt,[1] namely, the directory of Bell Laboratories with approximately 18,000 entries.

4.1 Implementation

In the implementation of the method, we have partitioned the alphabet into only two letter classes. This results in $2^8 = 256$ string classes, which is a suitable number for a directory of the size we were working with. The letter classes, class 0 and class 1, have been chosen as shown in Table III. Each class contains the same number of letters, and easily confusable letters, such as "B," "D," "P," or "A," "J," "K," are always within the same class. A dummy letter "_," denoting an empty position, has been included to fill up names with less than the maximum permissible number of letters in order to allow a uniform treatment of all names. As a consequence of using only two letter

Table III—Equivalence classes of the alphabet

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

classes, not all letters within a class are easily confusable. This, however, does not have any effect on the efficiency of the method, since it is only important that easily confusable letters be in the same class.

The classification of a name into a string class has been performed by assigning the letter classes according to Table III and evaluating the resulting binary number. As an example, this procedure classifies the name SMITH CC into string class 236 according to

$$\text{SMITH CC} \rightarrow 11101100_2 = 236_{10}.$$

Following this scheme, the existing alphabetically ordered directory has been reorganized. All entries have been classified and stored in the disk space provided for the corresponding string class, with the blocks being stored sequentially in order of increasing numbers. The blocks varied in size from 0 through 328, with an average of 70 names per block. This distribution was viewed as satisfactory, since only 10 percent of the blocks had more than 160 names. In addition to the directory entries, a table is stored in the directory file which, for each block, contains its size and the address of its first member. Using this table, an entire directory block can rapidly be read into primary memory during the course of a single access to secondary storage. In our implementation on a Data General ECLIPSE minicomputer, the time to examine one directory block, that is, to determine the entry with the minimum distance in that block, is approximately 0.4 second.

To increase the convenience of application, the search program allows for specifying only the first of two initials, even if both initials are listed in the directory. This option has been implemented by extending the distance matrix to the position of the second initial, but assigning zero distances to all letters in this position. For the retrieval of the name this means that the second initial is considered irrelevant. It is obvious, however, that, since less information is provided by the user, the rate of erroneous retrieval will be higher. Also, since two letter classes in the missing position have to be taken into account in the generation of string classes, the search time will be increased. In the worst case, the number of string classes to be searched will be doubled. However, this option does not have any effect on the retrieval in the case where both initials are specified.

4.2 Experimental procedure and results

For the experimental evaluation of the method, we used the same test data as in Ref. 1. These were 50 randomly chosen names from the directory of Bell Laboratories processed using speaker-dependent reference patterns in the acoustic analyzer. Each name was spoken by ten speakers, six male and four female. In addition, we tested the same

Table IV—Accuracy results of the experimental investigation

Speaker	Errors in 50 Names	
	Speaker Independent Patterns	Speaker Dependent Patterns
1	1	0
2	2	0
3	0	1
4	1	2
5	4	3
6	1	0
7	0	1
8	0	0
9	5	0
10	0	0

data using speaker-independent reference patterns to evaluate the performance of our system with an input of higher error rate. For the purpose of this experiment, all initials, as listed in the directory, were spoken, and it was assumed in the evaluation that the "stop" command was always correctly recognized.

The string errors are summarized in Table IV for each of the speakers and both types of reference patterns. The total number of errors is 14 for the speaker-independent patterns and 7 for the speaker-dependent patterns, corresponding to average error rates of 2.8 and 1.4 percent, respectively. Compared to the results reported in Ref. 1, the present average error rate is about a factor of 4 smaller. It is interesting to note that in 9 out of the 21 total errors, the last name was correctly found, and only the initials had been confused. Six of these cases affected the same name, with "G" always being recognized instead of "T."

It should be emphasized that all errors are a consequence of confusion in the acoustic analyzer alone and are in no way caused by a failing of the search technique. Any search strategy guaranteeing retrieval of the minimum-distance directory name would have yielded the same results. An example of incorrect recognition, in which the spoken name does not have the minimum distance of all names, is illustrated in Table V. On the other hand, note that a relatively high value of the total distance of a retrieved name does not preclude correct recognition. This is illustrated by the example in Table VI, which shows a distance matrix from which the spoken name was correctly recognized, although some of the letters occur in a very high ranking position.

The effectiveness of the search method, given in terms of the number of classes examined, is demonstrated in Fig. 1. The curves were

obtained by averaging the values from the speaker-independent and speaker-dependent trials. With speaker-independent reference patterns, about twice as many classes had to be searched as with speaker-dependent patterns. Curve (a) depicts the number of classes that were examined to meet the stopping criterion, that is, to guarantee the retrieval of the name with the minimum distance. On the average, only 2 percent of the whole directory had to be examined. This corresponds to an average search time of only 2 seconds in our implementation.

Curve (b) of Fig. 1 depicts the number of classes that had to be examined until the name with the minimum distance was first encountered, without attention to the stopping criterion. The significance of this curve is that it describes the effect that a limitation on the number of classes searched would have. For example, a limitation to only one class would still result in the retrieval of the name with the minimum distance in about 86 percent of the cases, and a limitation to 25 classes would give 99.9 percent. This latter possibility is of special interest, as it accelerates the search in the most time-consuming cases while having only negligible effect on the accuracy.

Furthermore, curve (b) of Fig. 1 reflects the confusability between the two letter classes. The fact that 86 percent of all names were encountered in the first string class examined means that no class confusion had occurred in any of the letter positions. Since a name consisted of seven letters on the average, it follows that, for the present distribution of letters, the probability of confusion between the two letter classes is about 0.02 (which could, of course, have also been measured directly).

Though this already seems to be a small value, a reduction to zero would result in a substantial reduction in the search, for, with zero confusability between the two letter classes, the search could be successfully terminated after only one string class had been examined.

Table V—Interleaved *L* and *D* matrices for an example of incorrect recognition. Spoken: TATE BA, total distance 1.71 (solid circles). Recognized: CATT GA, total distance 1.62 (dashed circles).

(T)	0.25	(A)	0.25	G	0.21	G	0.22	V	0.24	C	0.26
(C)	0.26	K	0.25	C	0.24	(E)	0.27	(G)	0.24	(A)	0.28
G	0.29	J	0.30	(T)	0.25	D	0.28	T	0.25	K	0.29
V	0.31	C	0.33	V	0.29	V	0.31	D	0.30	J	0.33
Z	0.31	Z	0.43	D	0.32	B	0.31	C	0.36	Z	0.41
P	0.38	T	0.50	Z	0.34	(T)	0.34	Z	0.36	I	0.50
D	0.39	I	0.52	E	0.40	P	0.42	P	0.37	W	0.55
B	0.51	V	0.55	P	0.42	C	0.42	E	0.39	T	0.56
A	0.51	W	0.58	B	0.46	Z	0.44	(B)	0.41	S	0.56

Table VI—Interleaved *L* and *D* matrices for an example of correct recognition Spoken: VIROST AM, total distance 3.93.

T 0.31	(I) 0.58	(R) 0.52	E 0.41	(S) 0.27	G 0.30	G 0.37	(M) 0.58
E 0.33	Y 0.67	M 0.75	A 0.42	F 0.39	E 0.40	K 0.37	L 0.65
G 0.33	K 0.75	F 0.87	P 0.42	X 0.42	(T) 0.40	B 0.38	U 0.69
D 0.35	W 0.79	L 0.89	D 0.43	M 0.56	C 0.41	C 0.41	N 0.71
C 0.38	R 0.83	I 0.96	V 0.46	N 0.56	K 0.43	Z 0.43	K 0.73
B 0.38	A 0.84	Y 0.98	K 0.477	K 0.64	P 0.45	D 0.44	Q 0.74
P 0.38	N 0.84	N 1.1	T 0.48	H 0.68	D 0.46	V 0.45	F 0.77
Z 0.40	F 0.88	S 1.1	Z 0.50	Ll k0.68	Z 0.47	T 0.46	S 0.77
(V) 0.45	C 0.89	A 1.1	J 0.51	J 0.69	B 0.49	J 0.47	X 0.81
K 0.45	M 0.89	B 1.1	G 0.52	C 0.71	V 0.49	(A) 0.49	J 0.83
A 0.47	J 0.89	C 1.1	B 0.56	A 0.72	A 0.56	P 0.50	C 0.83
J 0.49	L 0.92	D 1.1	Q 0.62	V 0.72	J 0.58	E 0.51	V 0.92
H 0.59	O 0.93	E 1.1	(O) 0.64	W 0.74	H 0.69	N 0.70	P 0.96

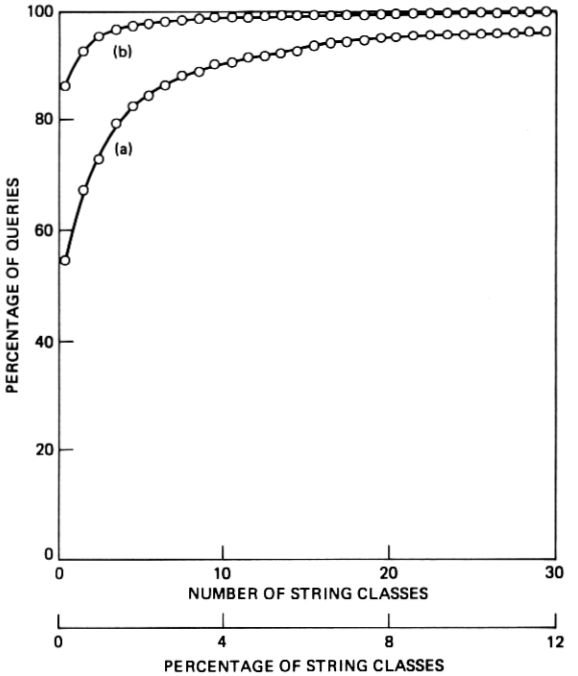

Fig. 1—Percentage of queries satisfied by examining a given number (percentage) of directory classes when the query is satisfied. (a) By meeting the stopping criterion. (b) By encountering the minimum-distance string.

Thus, on the average, only 0.4 percent of the whole directory would have to be examined, an improvement by a factor of 5 over the present performance. An improvement of the acoustic analyzer, even if only concerning the confusability between the two letter classes, would therefore be desirable.

V. CONCLUSION

The problem of automatically obtaining directory information from spoken spelled names is made difficult by the high degree of acoustic similarity among many of the letters of the alphabet. This results in unreliable identification of letters by the acoustic analyzer. We have described a method that provides speedy, highly correct recognition of names, even when driven by an acoustic analyzer with a significant error rate. Accuracy is achieved by using the minimum-distance criterion for selecting entries from the directory. Speed is obtained by partitioning the directory according to an equivalence relation which stores together all names that are acoustically similar on a letter-by-letter basis. The examination of directory blocks in order of increasing

distance makes it possible, on the average, to find the minimum-distance directory entry after examining only a fraction of the whole directory. A stopping criterion guarantees that the name found truly is the minimum-distance name in the directory.

The method has been tested using a directory with 18,000 entries and two different types of reference templates. With speaker-dependent reference patterns, which provided 80 percent correct recognition of individual letters, the correct name was found in 98.6 percent of the queries. On the average, only 1.2 percent of the directory file had to be examined per query. With speaker-independent reference patterns, which gave 71 percent correct recognition of individual letters, the name recognition rate was 97.2 percent, and 2.8 percent of the directory file had to be examined.

The search technique described is not limited to processing acoustic input. It is also applicable to other pattern recognition problems where search patterns are composites whose individual components are unreliably specified; for example, determining the words in a document that has been scanned by an optical character recognition device. The only requirement for our technique to be applicable is that some natural equivalence relation apply to component values. The performance of the technique will then be largely determined by the frequency with which inequivalent values are substituted for the intended values. If no confusion occurs between inequivalent values, the minimum-distance entry will be in the very first block examined, whereas a high incidence of confusion will lead to a corresponding degradation of performance. It should be obvious that the same equivalence relation need not be used for every component of a query, nor do the components even have to range over the same domain of values.

VI. ACKNOWLEDGMENT

The authors thank L. R. Rabiner, A. E. Rosenberg, and J. G. Wilpon, who provided the acoustic analyzer and test data used in the experimental evaluation of the search method. They would also like to thank their colleagues for their helpful comments on this work.

REFERENCES

1. A. E. Rosenberg and C. E. Schmidt, "Automatic Recognition of Spoken Spelled Names for Obtaining Directory Listings," B.S.T.J., *58,* No. 8 (October 1979), pp. 1797–1823.
2. F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. Acoustics Speech and Signal Processing, *ASSP-23* (1975), pp. 67–72.
3. A. V. Aho, T. G. Szymanski, and M. Yannakakis, "Enumerating the Cartesian Product of Ordered Sets," Proc. 14th Annual Conference on Information Sciences and Systems, March, 1980.
4. 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. 571–592.