Speech Recognition Experiments with Linear Predication, Bandpass Filtering, and Dynamic Programming

GEORGE M. WHITE, MEMBER, IEEE, AND RICHARD B. NEELY, MEMBER, IEEE

Abstract—**Automatic speech recognition experiments are described in which several popular preprocessing and classification strategies are compared. Preprocessing is done either by linear predictive analysis or by bandpass filtering. The two approaches are shown to produce similar recognition scores. The classifier uses either linear time stretching or dynamic programming to achieve time alignment. It is shown that dynamic programming is of major importance for recognition of polysyllabic words. The speech is compressed into a quasi-phoneme character string or preserved uncompressed. Best results are obtained with uncompressed data, using nonlinear time registration for multisyllabic words.**

I. Introduction

THIS paper discusses techniques used in automatic recognition of isolated utterances. The development of isolated word recognition systems is a challenging problem that can have a significant economical impact and can also provide acoustic analysis components for continuous speech recognition systems.

In this paper we select several of the more promising techniques described in the literature of isolated word recognition and combine and compare them in a set of controlled experiments. This is apparently the first published comparative study of its kind in recent years. One combination of techniques yields some of the highest recognition scores yet achieved for English vocabularies of 90 words or more. We combine or compare linear predictive coding [3] with bandpass filtering; dynamic programming [6] with linear time normalization; and "character-string encoding" (C.S. Code) [5] with a simple bandpass filter space representation of speech. The results of the comparisons are given in five tables in Section IV. The tables also report the effect on system performance of 1) vocabulary, 2) parametric representation, 3) time alignment, 4) C.S.Code and 5) a combination of time normalization and C.S.Code. Our results show that dynamic programming, which achieves nonlinear time scaling, is extremely useful for automatic speech recognition of multisyllabic utterances.

Nearly all successful isolated utterance recognition systems of today use utterance prototypes. An utterance "prototype" is a reference utterance against which unknown utterances are compared. Prototypes, or templates as they are sometimes called, are represented by a time sequence of acoustic "features." These features may be simply the output of a set of filters or they may be the output of complicated "feature extractors." We use linear predictive coding and bandpass filtering to provide features for representing utterance prototypes. In some cases, we then apply a data compression operation called C.S. Code which is a mapping of utterances into a string of quasi-phonemic labels. Comparison of an unknown utterance and a prototype takes place by aligning utterances and prototypes in time and then measuring the degree of similarity between sounds in the same time unit. We tested two different time alignment strategies and several "similarity measures." Time alignment and similarity measurements are grouped together and called utterance "classification" strategies. The measurement of features is called "preprocessing." Details of the preprocessing and classification techniques are given in the next section.

Manuscript received February 28, 1975; revised April 11, 1975 and September 19, 1975.

The authors are with the Palo Alto Research Center, Xerox Corporation, Palo Alto, CA 94304.

II. Speech Recognition Algorithms

The speech recognition systems compared in this paper had three stages: a *preprocessing* stage followed by a *data reduction* stage followed by a *classification* stage. The preprocessing stage used *linear predictive coding* (LPC) or *bandpass filtering*. The data reduction stage used *character-string encoding* or no *C.S.Code*. The classification stage used *linear time scaling* or *dynamic programming nonlinear time scaling* (Figs. 1-5).

Speech sound similarity is defined to be proportional to the distance separating two sounds in a vector space defined by bandpass filter parameters, linear predictive coefficients, or character strings. Vector spaces of 6 and 20 dimensions are defined by 6 and 20 channel filter banks. Linear predictive coding establishes a vector space of dimensionality equal to the number of predictive coefficients. Similarity measurement with *C.S.Code* takes place by measuring distances between reference points rather than the actual points.

To understand C.S.Code, consider the following. A succession of points in vector space is produced by successive time samples of speech. Each point is given a quasi-phonemic label. Each label is taken from a nearby reference point. (The reference points are established during a prerecognition "learning" stage during which about 20 steady-state speech sounds are encoded as reference points.) Establishing reference points for C.S.Code is the subject of previous publications [4], [5]. The succession of quasi-phonemic labels produced in this way are saved to represent speech and the original spectral information is discarded. Each reference point is taken to be an approximation to an original spectral point. In "matching" the spectra of two utterances that have been character-string encoded in this way, distances are measured between reference points. Encoding utterances as character strings introduces some class-

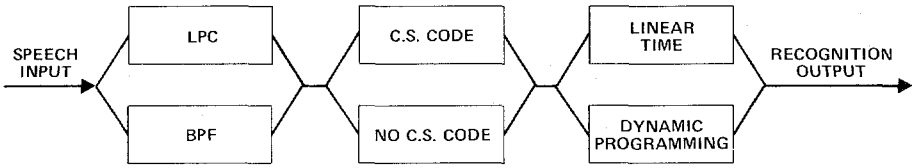

Fig. 1. Components of speech recognition system that were tested.

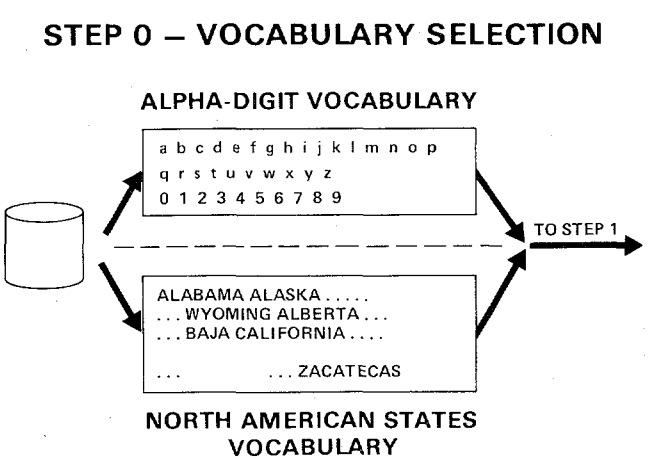

Fig. 2. Vocabulary selection. Two different vocabularies were selected from disc storage.

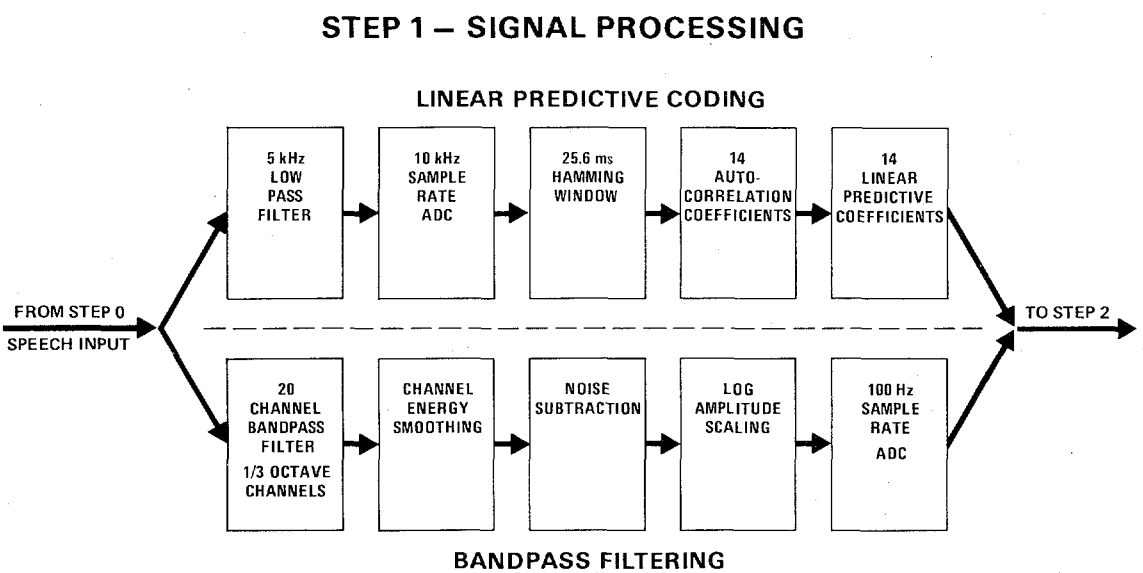

Fig. 3. Signal processing. Either LPC or bandpass filtering were used.

ification errors but it also greatly reduces the time and storage needed to process speech.

For the *bandpass filtering preprocessing* method, the *speech sound similarity measure* used was the Chebyshev norm (absolute value norm). This meant summing the absolute value of the differences of the coordinate values between the unknown and template every 10 ms. (We obtained slightly better results with the Euclidean norm, but it was not used because it was more expensive computationally. A similar result was obtained by Neroth [1] who compared the Chebyshev and Euclidean norms.) Before measuring distances, all filter samples were normalized by dividing by the total energy. (This normalization technique was previously used by Shearme and Leach [2].)

The *LPC preprocessing* method employed a *speech sound similarity measure* based on Itakura's linear predictive residual. A detailed description of the linear predictive residual is found in a paper by Itakura [3].

Of the two *time alignment methods*, linear time scaling is the simplest. *Linear time scaling* means that two utterance representations to be compared are stretched or compressed *linearly* so that they become the same length. Time normalized utterances are also shifted relative to one another in order to overcome misalignment due to poor detection of the beginnings and endings of utterances. However, nonlinear time translations within the interior of an utterance will still cause interior mismatch. *Dynamic programming*, however, does allow for nonlinear translation, and thus achieves better interior match-

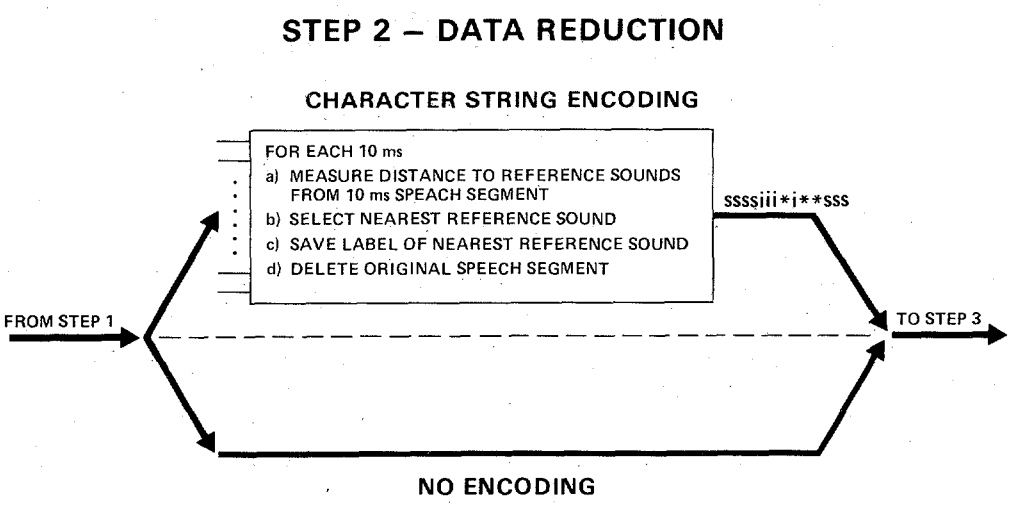

Fig. 4. Data reduction. The speech data sampled from Step 1 were either encoded as a character string of subphonemes *or* they were *not* encoded.

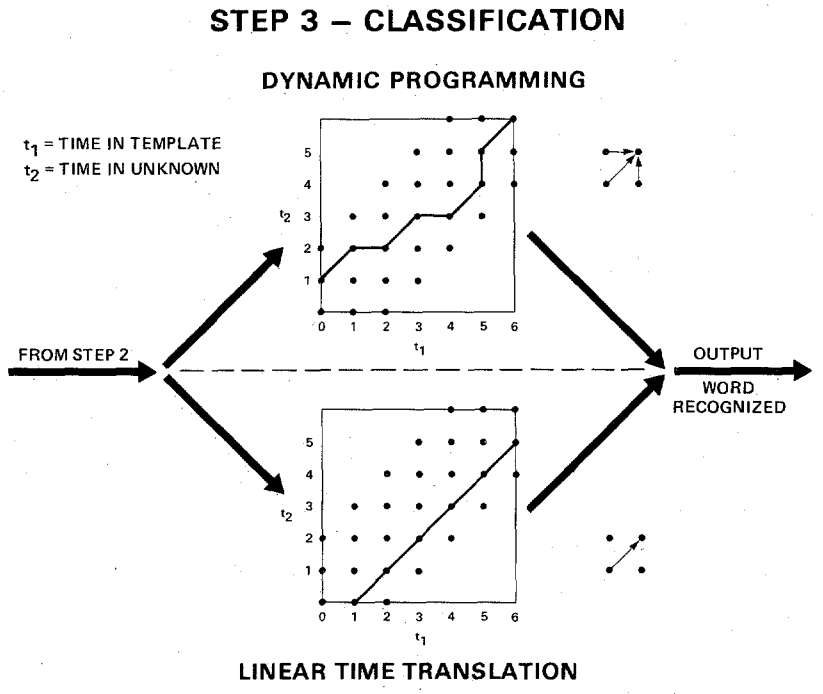

Fig. 5. Time alignment strategies. Either linear time translation or dynamic programming was used to align an unknown utterance with a template.

ing. Dynamic programming requires more computation, but significantly improves recognition results for multisyllabic utterances.

The dynamic programming equation used was

$$D_{ij} = d_{ij} + \min \{D_{ij-1}, D_{i-1j}, D_{i-1j-1}\}$$

where d_{ij} is defined to be the distance between the first utterance at time slice i and the second utterance at time slice j. D_{ij} is the total distance between the first and second utterances from their beginnings up to and including times i and j. The operation "min $\{a, b, c\}$" selects the smallest number from the set of numbers a, b, and c. For a more thorough discussion, see Itakura [3] or Sakoe and Chiba [6].

III. Experimental Conditions

All speech used in these experiments was recorded on analog tape. The tapes are available from the authors. A Telex noise canceling microphone model number CS-75 was used. The recordings were made in a laboratory room with a 65 dB(A) noise level from an air conditioner. Two different vocabularies were recorded. The Alpha-Digit vocabulary was spoken by G. White. It contains the names of the letters of the alphabet and the digits zero through nine for a total of 36 words. (The alphabet was spoken as "A, Bee, Cee, . . ." and not "Alpha, Bravo, Charlie, ...") This vocabulary was spoken five times over the period of two days. The other vocabulary, the North American States vocabulary, contains 91 names: the 50 states of the United States, plus 10 Canadian providences, plus 31 Mexican states. It was pronounced five times by R. Neely in one day in five different recording sessions.

A data base of digitized speech was prepared from analog tapes using either LPC or filter bank analysis. The parameters used to specify the LPC operation and the filter bank opera-

tion are as follows: LPC analysis used a 10-kHz sampling rate, 8-bit resolution, 4 dB per octave preemphasis, a 5-kHz low-pass filter, and linear amplitude scaling. Fourteen LPC coefficients were calculated every 12.8 ms and a 25.6-ms Hamming window was used. The filter bank analysis used a 100-Hz sampling rate with amplitude averaging between samples, a logarithmic amplitude scale, 4 dB per octave preemphasis, 8-bit log resolution, and either six one octave filters or 20 one-third octave filters. The filters covered the frequency spectrum from about 100 Hz to 10 kHz. The details of the hardware used to perform these LPC and bandpass filtering operations are given in the following paragraph.

Speech from an analog tape was passed through a Hewlett-Packard analog filter set model 8056A which is a set of 25 one-third octave Chebyshev filters. Spectral shaping (preemphasis) was obtained by adjusting the gain of each of the one-third octave filters to obtain an average gain of about 4 dB per octave about 1000 Hz. In the LPC experiment, speech was low-pass filtered to remove all frequencies above 5 kHz. The low-pass filter was obtained by adjusting the gains on the HP filter channels to −40 dB for the frequencies above 5 kHz. In LPC analysis, the output of the filter bank was passed directly to a 13-bit analog-to-digital converter (ADC). (Only 8 bits of resolution were actually used of the 13 bits possible.) In bandpass filter analysis, the output of the filter bank was passed to a bank of rectifier-integrator circuits and then to the ADC. (Each "rectifier-integrator" circuit rectifies the voltage from a filter channel, subtracts a constant voltage to remove effects of background acoustic noise, sums the resulting voltages for 10 ms, and finally converts the output voltages from linear to logarithmic scales.) The ADC then performed the analog-to-digital conversion and passed the resulting 8-bit numbers to a minicomputer. The digital data was stored on a disc to create a data base for our recognition experiments.

The five repretitions of the Alpha-Digit vocabulary and of the North American States vocabulary were used in recognition experiments to obtain 20 different vocabulary tests. A single repetition was used to "train" the recognizer. (Training consisted of simply storing one example of each word in the vocabulary as an utterance prototype.) The remaining four repetitions were then used to supply unknown utterances to test the accuracy of the recognizer. Next a different repetition from the original five was selected to generate prototypes and again the other four were used to test the recognizer. This process of selecting one repetition from five and testing on the remaining four was repeated five times thus creating 5×4 vocabulary tests. For the 36 word Alpha-Digit vocabulary, this resulted in 720 individual utterance tests. Similarly, 1820 utterance tests were obtained from the North American States vocabulary.

IV. Results

In the four tables below, each "percentage correct" was based on 1820 or 720 utterances depending on which vocabulary was used. The North American States vocabulary had 1820 utterance tests and the Alpha-Digit had 720 utterance tests.

For test 1 in Table I, preprocessing was done by a 20 channel filter bank, with $\frac{1}{3}$ octave bandwidth per channel; time alignment was done by dynamic programming; the percentage correct was 98 percent for the Alpha-Digit vocabulary and was 99.6 percent for the North American States vocabulary. In test 2, which differs from test 1 only in the time alignment strategy, the "linear time" normalization was achieved as follows. All utterances were stretched or compressed linearly to be the same length, namely 50 time units long. Reference utterances (prototypes) were compared to unknown utterances by summing up the 50 distances between unknown and reference sounds in the same time units. Then unknown utterances were shifted linearly right and left relative to the prototype and the total distance recalculated. The smallest "total distance" was assumed to be the result of the proper time alignment and was taken as the proper measure of utterance and prototype similarity.

An experiment using LPC recently reported by Itakura [3] shows an error rate of 11.4 percent on the Alpha-Digit vocabulary which is to be compared to our 3 percent error rate for the same vocabulary (Table II). Since the LPC technique we employed was essentially a replication of Itakura's system, the difference in scores requires an explanation. The differences between Itakura's system and ours arises primarily from the bandwidth difference for speech input. Itakura used essentially six LPC coefficients and a 6-kHz sampling rate for his recognition of speech input over a 3.0-kHz phone line. Our system used 14 LPC coefficients with speech input through an 8-kHz noise canceling microphone connection with a 10-kHz sampling rate. We also used a $\frac{1}{3}$ octave filter bank for spectral shaping to preemphasize high frequencies. We also used a slightly different (improved?) dynamic programming strategy (see Sakoe and Chiba [6]). We believe that the differences in scores arise essentially because our system has better access to high frequencies necessary to distinguish some of the words confused by the Itakura recognizer.

The most important result of the time alignment experiments (Table III) is an indication of the extraordinary recognition power of dynamic programming on multisyllabic utterances. A second result is that linear time normalization (with right and left shifting) is as good as dynamic programming on monosyllabic utterances.

The most interesting result shown in Table IV is that C.S.Code seems to reduce the advantage of using dynamic programming relative to linear time normalization.

V. Conclusions

The data base used in these experiments is relatively small. We ask the reader to bear in mind that these are preliminary conclusions.

One conclusion of our work is that *bandpass filtering* and *linear predictive coding* can provide approximately equivalent bases for measuring speech waveform similarity. In other words, a Chebyshev or Euclidean metric in a bandpass filter space is approximately equivalent to the log ratio linear predictive *residuals* in LPC space when typical LPC and filter bank parameters are used. The errors made by both LPC and filter bank recognition systems were nearly the same. Nearly all the errors arose from confusions between /b/ and /d/ and between /m/ and /n/. Our experiments have not concentrated on nasal sounds for which differences are most likely to be found be-

TABLE I
VOCABULARY EFFECTS

Test	Preprocessing Method	Time Alignment	Alpha-Digit Vocabulary (Percent Correct)	North American States Vocabulary
1	20 channel	Dynamic Programming	98 percent	99.6 percent
2	20 channel	Linear Shift	98 percent	90 percent

The Alpha-Digit vocabulary has 36 words, mostly monosyllabic. The North American States word list has 91 words, mostly polysyllabic. Test 1 uses 20 bandpass filter channels and dynamic programming, and gets recognition results of 98 percent correct for the Alpha-Digit vocabulary and 99.6 percent correct for the North American States. The larger vocabulary gets a significantly higher score. But in test 2 it gets a significantly lower score. Dynamic programming is not used in test 2. Linear time shifting (translation) is used in test 2. Thus the vocabulary used has a profound effect on the apparent advantage of different time alignment strategies.

TABLE II
PARAMETRIC REPRESENTATION EFFECTS
(PREPROCESSING EFFECTS)

Preprocessing Method	Time Alignment Method	Alpha-Digit Vocabulary (Percent Correct)	Recognition Time per Utterance	Data Rate Bits/s Approximate
20 channel	Dynamic Programming	98 percent	30 s	12 000
LPC	Dynamic Programming	97 percent	20 s	4200
6 channel	Dynamic Programming	96 percent	15 s	3600
C.S.Code	Dynamic	91 percent	2 s	500

Preprocessing produces four different parametric representations which are arranged in order of increasing data compression (lower bit rate). The recognition accuracy goes down as compression goes up. The similar results for LPC and bandpass filtering shows that they are essentially equivalent.

TABLE III
TIME ALIGNMENT EFFECTS

Preprocessing Method	Time Alignment Method	Vocabulary	Recognition Accuracy (Percent Correct)
20 channel	Dynamic Programming	North American	99.6 percent
20 channel	Linear Shift	North American	90 percent
20 channel	Dynamic Programming	Alpha-Digit	98 percent
20 channel	Linear Shift	Alpha-Digit	98 percent

Time alignment methods include linear time shifting and dynamic programming. Dynamic programming produces nonlinear time warping to achieve the best match between template and unknown utterances. Note that dynamic programming achieves 99.6 percent correct on the multisyllabic words of the North American states word list while linear time normalization achieves only 90 percent. Note, however, that there is no advantage for monosyllabic words of the alphabet plus digits.

tween the LPC and bandpass filtering approach. But with this exception, our work builds confidence in the similar power of the two approaches.

A second conclusion is that popular data reduction techniques may be damaging to the performance of recognition systems. Table II shows preprocessing techniques ranked according to degree of data compression. A strong correlation to recognition accuracy is evident. Although we do not claim to have proved it in this paper, these experiments suggest that standard data reduction techniques have significantly reduced accuracies in computer recognition of speech. We suggest that progress in speech recognition will be aided by special purpose data processors capable of dealing with significantly higher data rates than general purpose computers. This would be one way to avoid the crippling effects of present data compression "preprocessors." It would also promote the use of dynamic time alignment schemes such as offered by dynamic programming which require extra computation.

TABLE IV
C.S.CODE INTERACTION WITH TIME ALIGNMENT

Preprocessing Method	Time Alignment Method	Vocabulary	Recognition Accuracy (Percent Correct)
20 channel	Linear Time	Alpha-Digit	94 percent
20 channel	Dynamic Programming	Alpha-Digit	98 percent
C.S.Code	Linear Time	Alpha-Digit	89 percent
C.S.Code	Dynamic Programming	Alpha-Digit	91 percent

In going from linear time alignment to dynamic programming for 20 channel representation, there is an increase in accuracy from 94 percent to 98 percent. However, with C.S.Code, the accuracy only goes from 89 percent to 91 percent. It appears that C.S.Code reduces the advantage of dynamic programming over linear time normalization. Note that "linear shift" and "linear time" differ in that "linear shift" tries several "linear time" alignments and preserves the best one.

The importance of vocabulary in evaluating recognition systems was demonstrated in our experiments. Japanese recognition scores have been higher than American scores for the past several years. Our results lend support to the idea that the difference in scores may be due to language and vocabulary differences. In fact, we originally chose the North American States vocabulary in order to test this idea. Itakura [3] had reported a very good score, 97.3 percent correct on 200 Japanese geographical names. Each name had an average of 3.5 syllables. However, Itakura reported only 88.6 percent correct for the English alphabet plus digits (Alpha-Digit vocabulary). This led us to suppose that the differences in scores were largely due to vocabulary differences. In order to minimize the vocabulary differences, we looked for a similar multisyllabic word list and came up with the North American States vocabulary. Our results showed a large "vocabulary effect."

In closing, we offer a general opinion on speech recognition research. General purpose computers are unable to process uncompressed speech with adequate speed for real-time speech recognition using utterance template matching techniques and comparing all templates composing the recognizer's vocabulary. In most speech recognition systems an attempt is made to overcome this problem through a special "preprocessing" step designed to reduce the amount of data that reaches the computer. This data reduction is usually so severe that relevant information is thrown away. This is a major cause of low

accuracy and lack of "robustness" of recognizers on most general purpose computers. Our research shows that relatively unsophisticated classification strategies perform very well if working with uncompressed speech data. The speech recognition research community would be well advised to pay more attention to those operations that it has come to call "preprocessing."

ACKNOWLEDGMENT

We thank D. R. Reddy for guidance and encouragement in our research. We also thank C. Hankins, P. Belew, and M. Wilmer for assistance in carrying out the experiments.

REFERENCES

[1] C. C. Neroth, "Audio graphic programming system," Ph.D. dissertation, Univ. California, Berkeley, pp. 41–45, 1972.
[2] J. N. Shearme and P. F. Leach, "Some experiments with a simple word recognition system," *IEEE Trans. Audio Electroacoust. (Special Issue on Speech Communication and Processing–Part II)*, vol. AU-16, pp. 256–261, Mar. 1968.
[3] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing (Special Issue on IEEE Symposium on Speech Recognition)*, vol. ASSP-23, pp. 67–72, Feb. 1975.
[4] G. M. White, "Speech recognition with character string encoding," in *Proc. 1972 IEEE Conf. Decision and Control*, New Orleans, LA, Dec. 13–15, 1972, TJ217,117.
[5] —, "Simple techniques for transforming speech to quasi-phoneme strings," in *Proc. Speech Communication Seminar*, Stockholm, Sweden, Aug. 1–3, 1974, pp. 225–231.
[6] H. Sakoe and S. Chiba, "A dynamic programming approach to continuous speech recognition," in *Proc. 7th Int. Congr. Acoustics*, 1971, Paper 20, p. C13.
[7] O. Fujimura, "Syllable as a unit of speech recognition," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, PA, Apr. 1974, pp. 148–153; also in *IEEE Trans. Acoust., Speech, Signal Processing (Special Issue on IEEE Symposium on Speech Recognition)*, vol. ASSP-23, pp. 82–87, Feb. 1975.

Correspondence

Utterance Classification Confidence in Automatic Speech Recognition

RALPH KIMBALL AND MICHAEL H. ROTHKOPF

Abstract—**A variety of automatic speech recognition experiments have been executed that support a measure of confidence for utterance classification. The confidence measure tested was the ratio of the two best "Hamming distance" scores obtained in matching utterance templates with an unknown utterance. The results show that it is possible to reliably predict when the utterance classifier has made the correct decision.**

INTRODUCTION

This correspondence describes a measure derived from experiments in automatic speech recognition that has reliably predicted when an utterance classifier has made the correct decision. The principal advantage of such a measure is that it provides a consistent scheme for deciding when to invoke a more costly, but more powerful, classifier.

DISCUSSION OF RESULTS

A common method of representing utterances for speech recognition is by coding them into a string of characters [1]. For instance, given a six-channel filter bank, a point in six-dimensional space is generated each time the speech waveform is sampled. Techniques of cluster analysis allow one to label any six-dimensional point with the label of the nearest cluster center. If the labels are letters, a character-string encoding of the complete utterance results. If the beginning and end of the utterance can be reliably identified, it is convenient to time normalize the utterance so that character strings will have the same length. Now the speech recognition task consists of correctly matching character strings.

Given an unknown string U and a set of k known strings $T_1, \cdots, T_k$, we wish to choose the known string T_M most similar to U. The jth character of the string T_m is denoted by T_{mj}. The Hamming distance H_m between T_m and U is the total number of character differences between the respective strings:

$$H_m = \sum_m (1 - \delta_{T_{mj}U_j}).$$

The minimum distance D_1 is

$$D_1 = \min_m [H_m] = H_M.$$

That value of m for which H_m is minimized, call it M, is taken to identify the unknown utterance.

The second smallest distance is

$$D_2 = \min_{m \neq M} [H_m].$$

The Hamming ratio $R = D_2/D_1$ is an indication of the degree of competition among alternative classifications. Figs. 1–3 are graphs of the cumulative probability function of R, i.e., $P(R) = \text{Prob}\ (R \leqslant R_0)$, encountered in experiments involving 54 utterances. An instance of each utterance not used to form the templates provided the set of "unknown utterances." Observe that all of the wrong classifications are for small values of R at the left side of the graph. This leads us to propose the "Hamming ratio" as a measure of classification confidence.

If the Hamming ratio R falls below a specific threshold R_T, the classification cannot be taken seriously, and alternate more sophisticated pattern matching schemes must be employed. Conversely, if $R > R_T$, we may assume that the classification is correct with high likelihood, obviating the use of more expensive techniques.

Manuscript received November 18, 1974; revised November 6, 1975.
The authors are with the Palo Alto Research Center, Xerox Corporation, Palo Alto, CA 94304.