Isolated and Connected Word Recognition—Theory and Selected Applications

LAWRENCE R. RABINER, FELLOW, IEEE, AND STEPHEN E. LEVINSON, MEMBER, IEEE

(Invited Paper)

Abstract—**The art and science of speech recognition have been advanced to the state where it is now possible to communicate reliably with a computer by speaking to it in a disciplined manner using a vocabulary of moderate size. It is the purpose of this paper to outline two aspects of speech-recognition research. First, we discuss word recognition as a classical pattern-recognition problem and show how some fundamental concepts of signal processing, information theory, and computer science can be combined to give us the capability of robust recognition of isolated words and simple connected word sequences. We then describe methods whereby these principles, augmented by modern theories of formal language and semantic analysis, can be used to study some of the more general problems in speech recognition. It is anticipated that these methods will ultimately lead to accurate mechanical recognition of fluent speech under certain controlled conditions.**

I. INTRODUCTION

ALTHOUGH a great deal has been learned about the fundamental processes of speech production and speech perception, the goal of mechanical recognition of fluent speech remains elusive [1]-[5]. Speech recognition, however, has made major strides forward in the past decade, and it has advanced to the point where several commercial systems are currently available [6]-[11]. These commercial systems are predominantly isolated word, speaker-trained systems which achieve word accuracies greater than 95 percent in noisy environments. At least one system, however, is a speaker-independent, isolated word recognizer operating over dialed-up telephone lines [7], while another is a speaker trained system that can handle a connected string of words (typically digits) [8].

In the laboratory, equally impressive advances have been recorded for speech recognition. A wide range of systems based on isolated words (both speaker-trained, and speaker-independent) have been developed for use over dialed-up telephone lines [12]-[16]; and more recently speaker-independent, connected-word systems have been proposed for connected digit recognition [8], [17]-[19].

As the capabilities of the word recognizers have improved, the tasks to which they have been applied have become more sophisticated, and more difficult. Such tasks have included chess playing, data retrieval and management, airlines information and reservations, and automatic recognition of read text extracted from patents on lasers [20]-[23]. Much of this advanced research has been under the auspices of ARPA, and excellent summaries of this work are available in [4] and [24].

This paper is intended to be a tutorial in the concepts and theories underlying modern speech-recognition systems, both practical and experimental. Two aspects of the subject are given special attention. First, we treat speech recognition as a classical problem in pattern recognition and show how some fundamental ideas from signal processing, information theory, and computer science can be utilized to provide the capability of robust recognition of isolated words and simple connected-word sequences. We then describe methods whereby these principles, augmented by modern theories of formal language and semantic analysis, can be combined to study some of the more general problems of speech recognition. In particular, we show how these theories can be applied to improve the recognition accuracy of a nonideal acoustic pattern recognizer. It is anticipated that these investigations will ultimately afford accurate mechanical recognition of fluent speech provided that it is part of a "task-oriented" dialog, that is, it is restricted to pertain to a well-defined, carefully circumscribed topic.

The outline of this paper is as follows. We begin, in Section II, with an overview of the pattern-recognition aspects of speech recognition. In this section, we are concerned with methods for short-time spectral estimation and elaborate on the filter bank and linear prediction methods. We also discuss other aspects of the pattern-recognition paradigm including similarity measures, temporal alignment of speech patterns, and statistical decision strategies. We then describe, within this framework, the basic word-recognition system, giving some details of its implementation, operation, and performance.

Section III provides a discussion of the application of pattern-recognition techniques to the construction of speech-recognition systems designed to perform specific tasks. In these examples, the special requirements of each task influence the way the general theories are utilized and adapted.

A relatively straightforward application of the basic principles allows us to build a voice-operated telephone repertory dialer. This particular system exploits some rudimentary task constraints by partitioning the vocabulary into sets of which only one is appropriate to any of the specific types of commands to which the system can respond.

A more sophisticated kind of constraint is used to build a telephone directory listing retrieval system. Here the constraints implicit in a telephone directory are used to actually correct acoustic-recognition errors. In this context, we intro-

Manuscript received October 17, 1980; revised January 8, 1981.
The authors are with Bell Laboratories, Murray Hill, NJ 07974.

0090-6778/81/0500-0621$00.75

duce the notion of list searching on an incomplete or corrupted key. Later we will formalize and generalize this approach.

Finally we describe a system which recognizes strings of connected digits. This system is substantially different from the other two in that it recognizes strings of words uttered without pauses between them. This is made possible by an important generalization of the temporal-alignment procedure discussed earlier.

Each of these three systems represents an advance toward the ultimate goal of speech-recognition research, human/machine conversational-speech communciation. Over the years, this goal has proven to be a most elusive one. Part of the reason for the difficulty lies in the fact that extrapolation of the pattern-recognition paradigm does not provide a sufficiently general model of the speech-communication process. Thus in Section IV, we go on to consider some other disciplines which can be used to analyze some phenomena of speech not encompassed by the pattern-recognition model.

We begin with a brief description of the human speech-communication process including both the vocal and auditory apparatus and an abstract definition of communication. We then outline some parts of the theories of formal languages and semantic analysis which can be applied to speech recognition. In particular, we elaborate on a simple formalization of grammar which both dramatically increases the versatility and robustness of our speech-recognition machines and provides insights into the role of linguistic structure in speech recognition.

Next, we show how these theories can be implemented and how they increase the capabilities of our experimental speech-recognition systems. We first return to the notion of list searching. We formalize this procedure and present two algorithms. This discussion provides natural motivation for the notion of maximum-likelihood parsing leading to two different methods for continuous speech recognition. Finally, we develop some semantic theories which, when combined with the formal language theoretic results, permit us to simulate, in an elementary way, the entire speech-communication process.

We conclude with an evaluation of our current knowledge and experimental results, some directions for future research, and a few predictions of our ultimate accomplishments.

Although we have strived to describe the theory and framework of an arbitrary word recognizer in this paper, most of the examples and specific applications are derived from the research performed at Bell Laboratories. The advantage of this strategy is that the authors are familiar with this work and can accurately describe the theoretical principles, the results, and the conclusions drawn from this research. The disadvantage is that we are forced to omit describing, in detail, the work of our colleagues both in the commercial world and at universities and other research laboratories. We have, however, endeavored to reference outside work whenever possible, especially as it relates to the general principles we describe.

II. PATTERN-RECOGNITION MODEL FOR SPEECH RECOGNITION

Fig. 1 shows the canonic pattern-recognition model used in the majority of isolated word speech-recognition systems. There

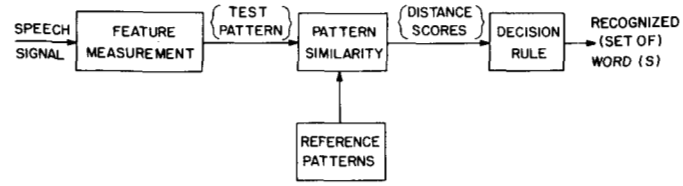

Fig. 1. Pattern-recognition model for speech recognition.

are three basic steps in the model:

1) feature measurement;
2) pattern similarity determination;
3) decision rule.

The input to the model is the acoustic waveform of the spoken input[1] (typically a word, or a connected string of words). The output of the model is a "best" estimate of the word (or words) in the input. Often the output of the model is a set of estimates of the words in the input, ordered by similarity, allowing the final decision of what was actually spoken to be deferred to a higher level of processing in the recognition system.

Before going into detail about how the three steps in the model of Fig. 1 are actually performed, it is worthwhile to make two observations about the model. The first point concerns the model itself. Although modifications have been made to one or more steps, no viable alternative to the model of Fig. 1 has been proposed. The second point explains why this model has been used for so long and for so many applications. The answer is that the model

1) is invariant to different speech vocabularies, users, feature sets, pattern similarity algorithms, and decision rules;
2) is easy to implement;
3) works well in practice.

Either of these reasons alone provides justification for use of the model; however, the combination provides compelling reasons for its use. Therefore we will be using and referring to the pattern-recognition model of Fig. 1 throughout this paper.

A. Feature Measurement

For the purposes of this paper, feature measurement is basically a data-reduction technique whereby a large number of data points (in this case samples of the speech waveform recorded at an appropriate sampling rate) are transformed into a smaller set of features which are equivalent in the sense that they faithfully describe the salient properties of the acoustic waveform. For speech signals, data-reduction rates from 10 to 100 are generally practical.

For representing speech signals, a number of different feature sets have been proposed ranging from simple sets such as energy and zero crossing rates (usually in selected frequency bands), to complex, "complete" representations such as the short-time spectrum, linear-predictive coding (LPC), and the homomorphic model [2]. For recognition systems, the moti-

[1] An important, real-world, problem with the simple model of Fig. 1 is that the problem of speech (or endpoint) detection (i.e., finding the signal in the given acoustic background) is ignored. We shall return to this problem later in this section.

vation for choosing one feature set over another is often complex and highly dependent on constraints imposed on the system (e.g., cost, speed, response time, computational complexity, etc.). Three of the most important of these criteria are

1) computation time
2) storage
3) ease of implementation.

Of course the ultimate criterion is overall system performance (i.e., accuracy with which the recognition task is performed). However, this criterion is a complicated function of all system variables.

To illustrate the techniques used in measuring features for speech recognition, we now discuss two frequently used feature sets, namely, the filter bank model and the LPC model.

1) Filter Bank-Analysis of Speech: One of the most popular set of features used for speech recognition is the output of a bank of filters, as shown in Fig. 2 [25]-[28]. Throughout this paper, we will assume the speech signal $s(n)$ is in digital form, i.e., it has been digitized at a sampling rate of F_0 samples per second. We will also assume that all signal processing is performed digitally. In many practical cases, however, analog signal processing is used to obtain the feature sets. This is especially true for the model of Fig. 2, since inexpensive implementations of this particular structure are fairly straightforward [6], [7], [9]-[11].

In filter bank-analysis model, the speech signal is passed through a bank of Q bandpass filters covering the speech band from about 100 Hz to some upper cutoff frequency (typically between 3000 and 8000 Hz). The number of filters used, Q, varies from about 5 to as many as 32, and the filter spacing is generally linear until about 1000 Hz, and logarithmic beyond 1000 Hz.

The output of each bandpass filter is generally passed through a nonlinearity (e.g., a square-law detector or a full-wave rectifier) and low-pass filtered to give a signal which is proportional to the energy of the speech signal in the band. A logarithmic compressor is generally used to reduce the dynamic range of the intensity signal and the compressed output is resampled (decimated) at a low rate (generally twice the low-pass filter cutoff) for efficiency of storage. At a given time sample m, the parallel outputs $x_1(m)$, $x_2(m)$, $\cdots$, $x_Q(m)$ define a Qth order feature vector $X(m)$. The time course of X defines a pattern, i.e.

$$X(m) = \{x_1(m), x_2(m), \cdots, x_Q(m)\} \tag{1a}$$

$$P = \{X(1), X(2), \cdots, X(M)\}. \tag{1b}$$

The pattern so defined is used for training or testing the recognition system.

At the bottom of Fig. 2, we show schematically a typical bandpass and low-pass filter. The cutoff frequencies of the bandpass filter are denoted as f_{L1} and f_{H1}. Typically, the filter bandwidth

$$\Delta f = f_{H1} - f_{L1} \tag{2}$$

is about 100 Hz for the low-cutoff filters, and as large as 500-1000 Hz for the high-band filters. Implementations that have been used are as simple as two-pole Butterworth filters, and as complex as 511 point, linear phase, FIR digital filters [25]-[28].

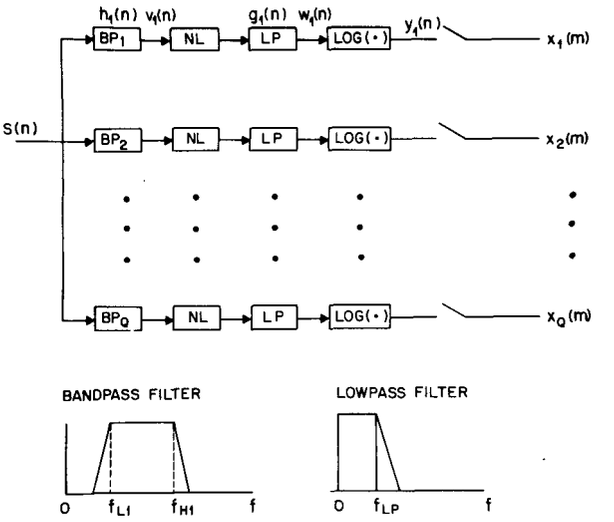

Fig. 2. Filter banks model for estimating recognition features from a bank of Q filters.

The low-pass filter cutoff frequency f_{LP} is typically about 20-30 Hz, implying output feature sampling rates of from 40 to 60 Hz. For the value of $Q = 5$, with a 40 Hz sampling rate, it is seen that a total of about 200 features are required to represent 1 s of speech. For 8000 Hz sampled speech, this represents a 40:1 reduction in data rate.[2]

It should be clear that each branch of the filter bank model of Fig. 2 is measuring (approximately) the speech energy in the band covered by the bandpass filter of the branch. For many recognition systems, this feature set is supplemented by adding a zero-crossing counter at the output of the bandpass filter, as shown in Fig. 3. (The zero-crossing count is, to a first approximation, a measure of the formant frequency for wide bandwidth filters.) By incorporating zero crossings into the system, the number of features is doubled with little increase in computation or system complexity. To the extent that zero crossing and energy are independent parameters in each band, the information obtained about the speech is increased at little cost. However, as we will see later, there is increased cost in the decision making parts of the recognizer. Hence the doubling of the number of features is generally unadvisable unless the features are truly independent and information bearing.

2) LPC Feature Model For Recognition: Another commonly used feature set for recognition is the LPC based feature set for recognition is the LPC based feature set originally proposed by Itakura [12].

The basic idea behind linear predictive coding is that a given speech sample can be approximated as a linear combination of

[2] It should be noted that a significant amount of information is lost from the signal when it is represented by 200 features per second, i.e., one could not synthesize a high-quality replica of the original signal from these features.

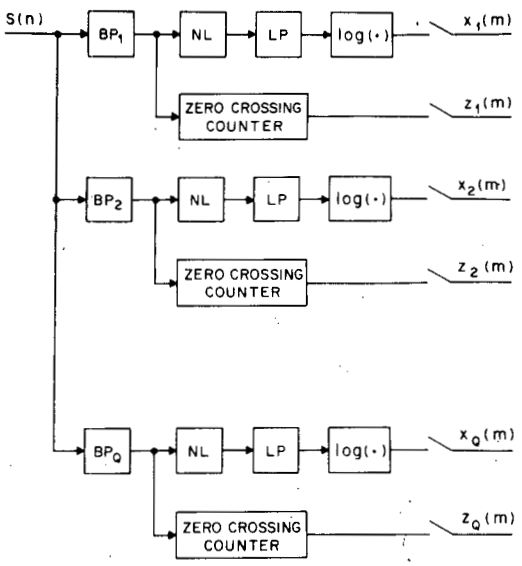

Fig. 3. The addition of zero-crossing measurements to the energy measurements of each band in the bank of filters model.

past speech samples. By minimizing the sum of the squared differences (over a finite interval) between the actual samples and the linearly predicted ones, a unique set of predictor coefficients can be determined. Linear-predictive coding can be readily shown to be closely related to the basic model of speech production in which the speech signal is modeled as the output of a linear, time-varying system excited by either quasi-periodic pulses (for voiced sounds) or random noise (for unvoiced sounds). The linear-predictive coding method provides a robust, reliable, and accurate method for estimating the parameters that characterize the linear, time-varying, system [2], [29], [30].

Fig. 4 shows a block diagram of the LPC-based feature analysis system. Unlike the bank of filters model, this system is a block processing model in which a frame of N samples of speech is processed, and a vector of features is measured. To obtain this vector, the speech signal is first preemphasized (to spectrally flatten the speech signal and to reduce computational instabilities associated with finite precision arithmetic [29]) using a fixed first-order digital system with transfer function

$$H(z) = 1 - az^{-1}, \qquad a = 0.95 \tag{3a}$$

giving the signal

$$\tilde{s}(n) = s(n) - as(n-1). \tag{3b}$$

The signal is next blocked into N sample sections (frames) for feature measurement. (Typical frame sizes are from 15 to 50 ms, i.e., $N = 150$ to 500 for a 10 kHz sampling rate.) Consecutive frames are spaced M samples apart. Clearly when $M < N$, there is overlap between adjacent frames. Such overlap inherently provides smoothing between vectors of feature coefficients. (Typical values of M are $M = N/3$, $M = N/2$, or $M = N$ for 3 to 1, 2 to 1, and no overlap, respectively.)

If we denote the lth frame of speech as $x_l(n)$, we have

$$x_l(n) = \tilde{s}(Ml + n), \qquad n = 0, 1, \cdots, N-1, \quad l = 0, 1, \cdots, L-1 \tag{4}$$

where $l = 0$ is the first frame and $l = L - 1$ is the Lth frame of speech. In order to minimize the effects of trying to analyze a slice of the speech waveform, a smoothing window $w(n)$ is applied to the data to taper the speech samples to zero at the end of the frame, giving the windowed signal

$$\tilde{x}_l(n) = x_l(n) \cdot w(n). \tag{5a}$$

A typical smoothing window used in LPC analysis systems is the Hamming window defined as

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right). \tag{5b}$$

The next step in the analysis is to perform an autocorrelation analysis of the windowed frame of data, giving

$$R_l(m) = \sum_{n=0}^{N-1-|m|} \tilde{x}_l(n)\tilde{x}_l(n+m), \qquad m = 0, 1, \cdots, p \tag{6}$$

where p is the order of the analysis system. (Typical values of p range from 8 to 12.) The feature set

$$X(l) = \{R_l(0), R_l(1), \cdots, R_l(p)\} \tag{7}$$

is often used as the analysis output of the LPC-based recognizer, since both reference and test patterns can be derived from the features of (7). However to perform both these tasks, a full LPC analysis is required in which a minimum mean squared error (in the time domain) all-pole fit to the spectrum of the frame of data is found by solving a set of linear simultaneous equations [29], [30]. The resulting all-pole model fit to the frame is of the form

$$A_l(z) = \frac{G}{1 + \sum_{m=1}^{p} a_l(m) z^{-m}} \tag{8a}$$

where G is a gain factor, and the set of coefficients $a_l(m)$, $m = 1, 2, \cdots, p$ define the all-pole model. If we define $a_l(0) = 1$, then we can define

$$A_l(z) = \frac{G}{\sum_{m=0}^{p} a_l(m) z^{-m}} \tag{8b}$$

and equivalently use the set $a_l(m)$ as the features for frame l.

To illustrate the LPC analysis method, Figs. 5 and 6 plots (for a 10 kHz sampling rate with $p = 14$) of

1) the windowed frame of data [part (a)];

2) the residual error (i.e., the signal which is not linearly predictable) by the model [part (b)];

3) the signal spectrum (as measured via an FFT) and the model spectrum [part (c)];

4) the spectrum of the error signal [part (d)] for the vowels /i/ (Fig. 5) and /a/ (Fig. 6). It can be seen from these figures

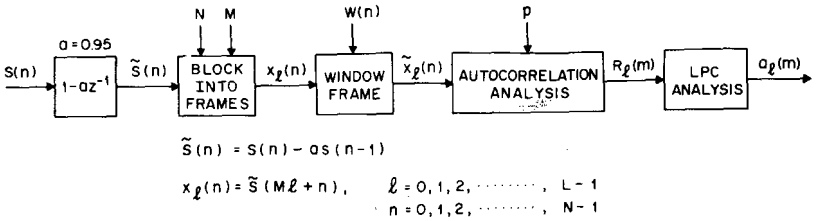

Fig. 4. Signal processing for extracting LPC features for recognition.

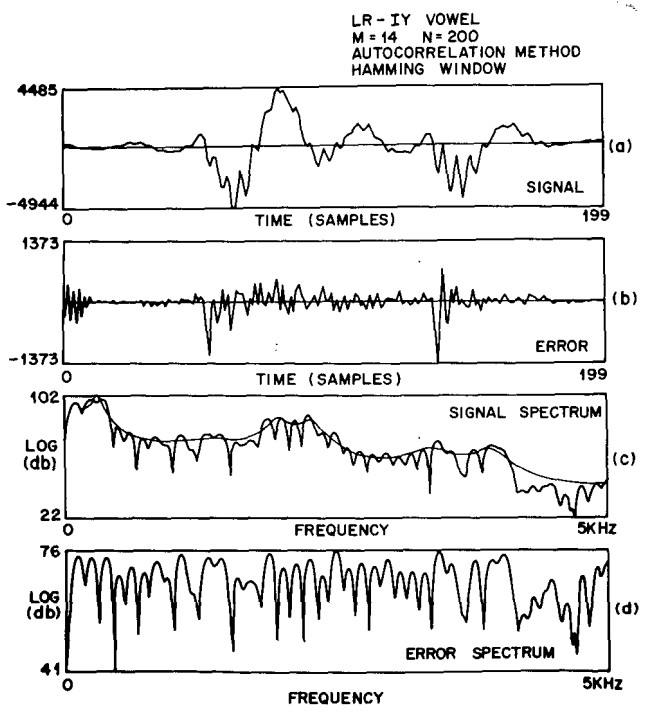

Fig. 5. Signals and spectra obtained from LPC model for vowel /i/.

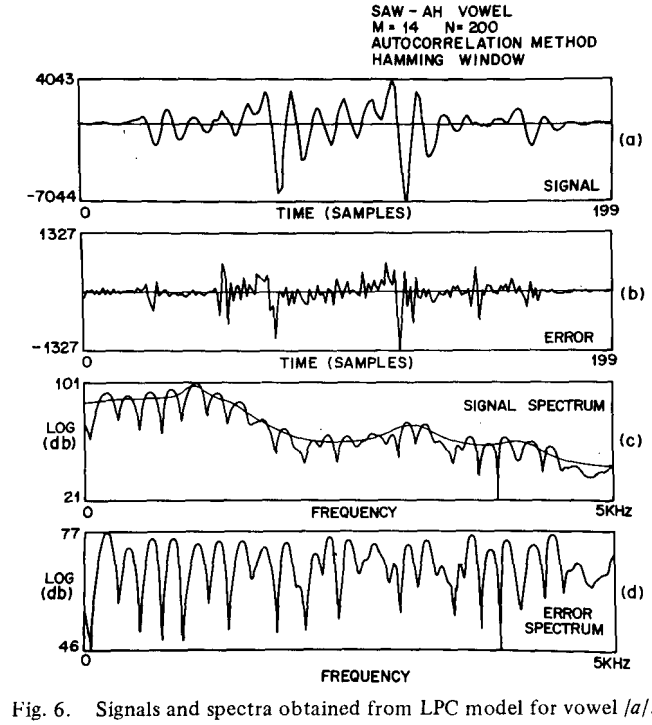

Fig. 6. Signals and spectra obtained from LPC model for vowel /a/.

that the LPC model provides a good fit to these simple vowel sounds.

B. Time Registration of Patterns

Once the patterns have been measured, the next step in the pattern-recognition model of Fig. 1 is to determine similarity between test and reference patterns. Because speaking rates vary greatly, pattern similarity involves both time alignment, and distance computation, and often these two are performed simultaneously.

Fig. 7 illustrates the function of time alignment between a test pattern $T(t)$ and a reference pattern $R(t)$ [12], [33], [35]. Our goal is to find an alignment function $w(t)$ which maps R onto the corresponding parts of T. The criterion for correspondence is that some measure of distance between the functions $D(T, R)$ be minimized by the mapping w. Thus, we seek $w(t)$ such that

$$D(T,R) = \min_{\{w(t)\}} \int_{t_0}^{t_1} d(t, w(t))G(t, w(t), \dot{w}(t))dt \qquad (9a)$$

where $\{w(t)\}$ is the set of all monotonically increasing, continuous differentiable functions; $\dot{w}(t)$ is the derivative of $w(t)$; $d(t, w(t))$ is a metric $d(T(t), R(w(t)))$ which is the pointwise distance from R to T; and G is a weighting function.

Unfortunately, the variational problem of (9a) is not in general solvable so we discretize the problem by letting

$$T = \{T(1), T(2), \cdots T(NT)\} \qquad (9b)$$

and

$$R = \{R(1), R(2), \cdots R(NR)\} \qquad (9c)$$

after which a number of techniques including the classical dynamic programming method may be used. The "optimum" time-alignment path is a curve relating the m time axis of the reference pattern to the n time axis of the test pattern, of the form

$$m = w(n). \qquad (10)$$

The constrained beginning and ending points of Fig. 7 can be formally expressed as constraints on $w(n)$ of the form

$$w(1) = 1 \qquad (11a)$$

$$w(NT) = NR. \qquad (11b)$$

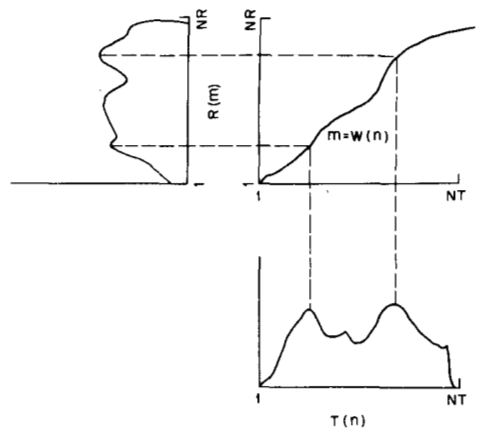

Fig. 7. Example of time registration of a test and a reference pattern.

Several techniques have been proposed for determining the alignment path w, including:

1) linear time alignment, i.e.

$$m = w(n) = (n-1)\frac{(NR-1)}{(NT-1)} + 1 \tag{12}$$

2) time event matching, i.e., times at which significant "events" occur in both reference and test patterns are found, and lined up in time

$$m_1 = w(n_1) \tag{13a}$$

$$m_2 = w(n_2) \tag{13b}$$

$$\vdots$$

$$m_Q = w(n_Q) \tag{13c}$$

and a functional fit to $w(n)$ is found based on these constraints. (Typically $w(n)$ is chosen to be a piecewise linear fit).

3) Correlation maximization, i.e., the warping function $w(n)$ is varied to maximize the correlation between reference and test patterns

$$R^* = \max_{w(n)} \sum_n (T(n)R(w(n))) \tag{14}$$

where the optimization is performed in a constrained manner.

4) Dynamic time warping, i.e., the warping curve is determined as the solution to the optimization problem

$$D^* = \min_{w(n)} \left[\sum_{n=1}^{NT} d(T(n), R(w(n))) \right] \tag{15}$$

where $d(T(n), R(w(n)))$ is the "distance" between frame n of the test pattern, and frame $w(n)$ of the reference pattern.

In the remainder of this section, we will develop the dynamic time-warping approach to registration of patterns since it has been shown to be extremely useful in a wide variety of speech-recognition systems [3], [8], [12], [15], [19], [23], [26], [31].

1) Frame-by-Frame Distance Measure: In order to implement the optimization of (15), the concept of distance between frames of features must be defined. Several possible distance measures can be used, depending on the form of the feature sets. For example, a simple Euclidean distance of the form

$$d(\boldsymbol{T}, \boldsymbol{R}) = \| \boldsymbol{T} - \boldsymbol{R} \| = \sum_{i=0}^{p} (T_i - R_i)^2 \tag{16}$$

where T_i and R_i are the ith components of the vectors $\boldsymbol{T}$ and $\boldsymbol{R}$, respectively, is often used.

Other distance measures which have been used include:

a) Covariance Weighting: The distance is defined as

$$d(\boldsymbol{T}, \boldsymbol{R}) = (\boldsymbol{T} - \boldsymbol{R})\tau^{-1}(\boldsymbol{T} - \boldsymbol{R})^t \tag{17}$$

where τ^{-1} is the inverse of the covariance matrix of features, i.e.

$$\tau = \begin{bmatrix} \overline{R_0R_0} - \bar{R}_0{}^2 & \overline{R_0R_1} - \bar{R}_0\bar{R}_1 \cdots \overline{R_0R_p} - \bar{R}_0\bar{R}_p \\ \vdots & \\ \overline{R_pR_0} - \bar{R}_p\bar{R}_0 & \overline{R_pR_p} - \bar{R}_p{}^2 \end{bmatrix} \tag{18}$$

where the overbar corresponds to expected value. This type of weighting compensates for correlation between features, and tends to give equal weight to all features in the overall distance.

b) Spectral Distance: For this measure the log spectra of reference and test patterns are obtained, and the distance is given as

$$d(\boldsymbol{T}, \boldsymbol{R}) = \int_\omega [\log [T(e^{j\omega})] - \log [R(e^{j\omega})]]^q \, d\omega \tag{19}$$

where q is usually an even integer (to make the qth power of the difference positive), and the integration is over the frequency range of interest. This distance measure has been shown to correspond well with subjective measures of difference, and several efficient techniques for approximating (19) have been proposed [32].

c) LPC Log Likelihood Measure: For feature sets based on LPC parameters, an extremely efficient distance measure was proposed by Itakura [12], of the form

$$d(\boldsymbol{T}, \boldsymbol{R}) = \log \left[\frac{\boldsymbol{a}_R \boldsymbol{V}_T \boldsymbol{a}_R{}^t}{\boldsymbol{a}_T \boldsymbol{V}_T \boldsymbol{a}_T{}^t} \right] \tag{20}$$

where $\boldsymbol{a}_R$ and $\boldsymbol{a}_T$ are the LPC coefficient vectors of the reference and test frames, and $\boldsymbol{V}_T$ is the matrix of autocorrelation coefficients of the test frame. An interpretation of the LPC distance of (20) is given in Fig. 8 in which the subscript R denotes reference, and the subscript T denotes test. The denominator of the term in brackets in (20) can be obtained by passing the test signal $S_T(n)$ through the inverse LPC system

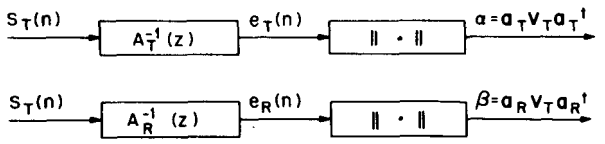

Fig. 8. Interpretation of LPC distance measure.

of the test, $A_T{}^{-1}(z)$ with response

$$A_T{}^{-1}(z) = \sum_{m=0}^{p} a_{mT} z^{-m} \tag{21}$$

giving the error signal

$$e_T(n) = \sum_{m=0}^{p} a_{mT} S_T(n-m) \tag{22}$$

and then by taking its energy giving

$$\alpha = \| e_T(n) \| = \sum_{n=0}^{N-1} [e_T(n)]^2. \tag{23}$$

Similarly, the numerator term can be obtained by passing the *same* test signal $S_T(n)$ through the inverse LPC system of the reference $A_R{}^{-1}(z)$ with response

$$A_R^{-1}(z) = \sum_{m=0}^{p} a_{mR} z^{-m} \tag{24}$$

giving the error signal

$$e_R(n) = \sum_{m=0}^{p} a_{mR} S_T(n-m) \tag{25}$$

with energy

$$\beta = \| e_R(n) \| = \sum_{n=0}^{N-1} [e_R(n)]^2. \tag{26}$$

Thus, from (20), (23), and (26) we get

$$d(T,R) = \log(\beta/\alpha) \tag{27}$$

showing that the distance between the two feature sets is related to the difference in LPC feature sets, which in turn is related to the differences in spectra between T and R frames. Markel and Gray [32] have quantified the interpretation of (20) as a spectral distance by showing it can be written as

$$d(T,R) = \log \left[\int_{-\pi}^{\pi} \left| \frac{A_R(e^{j\omega})}{A_T(e^{j\omega})} \right|^2 \frac{d\omega}{2\pi} \right] \tag{28}$$

i.e., an integrated square of the ratio of spectra between reference and test frames.

One of the most important aspects of any distance measure is the speed of computation, since distance calculations will be shown to be the most costly (time consuming) part of most recognition systems. As such any proposed distance measure which requires an inordinate amount of computation would not be a candidate for use in a practical system, no matter what its other advantages might be. On this basis, if we reexamine the four distance measures proposed in this section we get the following results:

1) Euclidean distance (16)—As proposed this distance measure requires $(p+1)$ multiplications, $(p+1)$ substractions, and $(p+1)$ additions. By using a sum of magnitudes rather than squares, the multiplications can be eliminated in the computation.

2) Covariance weighting (17)—This distance measure requires about $(p+1)^2$ multiplications and additions. For any reasonable size p, this computation is prohibitively large.

3) Spectral distance (19)—This distance requires the evaluation of an integral, or a discrete approximation to it. In either case the computation would be prohibitive; however Markel and Gray have proposed alternative measures which correlate well with (19), but with considerably reduced computation (e.g., the cosh measure) [32].

4) LPC distance (20)—As given in either (20), or as illustrated in Fig. 8, the computation of the LPC distance requires a large number of multiplications and additions. However Itakura [12] has shown that the distance d of (20) can be expressed (exactly) in the form

$$d(T,R) = \log \left[\sum_{m=0}^{p} \tilde{V}_T(m) R_R^a(m) \right] \tag{29}$$

where

$$V_T(m) = \frac{\tilde{V}_T(m)}{E_T} \tag{30}$$

where E_T is the normalized error of the LPC analysis of the test frame, and

$$R_R{}^a(m) = \sum_{k=0}^{p-m} a_R(k) a_R(k+m) \tag{31}$$

i.e., $R_R{}^a(m)$ is the autocorrelation of the finite vector

$$a_R = [1, a_R(1), a_R(2), \cdots, a_R(p)]. \tag{32}$$

Using (29), the computation of distance requires $(p+1)$ multiplications and additions, and one log.

These results show that both the Euclidean and LPC distances are reasonable candidates for distance measures for recognition systems and both these measures have been widely used in practice.

2) Dynamic Time-Warping Framework: As described earlier, the dynamic-time warping (DTW) alignment problem (15) is to find the "optimal" warping path $w(n)$ which minimizes the accumulated distance D between test and reference patterns, subject to a set of path and endpoint constraints. A variety of

formulations of the DTW problem, especially as applied to speech recognition, have been proposed by Sakoe and Chiba [33], and Myers *et al.* [34]. Rather than attempting to describe, or even summarize their work, we will just present a simple-minded approach that yields one DTW algorithm, [12], and then will briefly mention variants that have been proposed.

The basis behind most DTW algorithms is the realization that the solution to (15) is equivalent to finding the "best" path through a finite grid. As such, classical "path-finding" techniques could be used to solve the problem. In particular, it has been shown that a simple recursive technique could be used to find the best path in the grid. To see how this recursive solution is implemented, we must first define the minimum accumulated distance function $D_A(n, m)$ as the accumulated distance from the initial grid point $n = 1, m = 1$, to the grid point (n, m). If we assume that n represents the independent grid search variable, and that all valid paths through the grid correspond to monotonically increasing time indices, then we can write the recursion

$$D_A(n,m) = d(T_n, R_m) + \min_{q \leq m} [D_A(n-1, q)]. \tag{33}$$

As illustrated in Fig. 9, (33) says that the minimum accumulated distance to the grid point (n, m) consists of the local distance d between feature sets T_n and R_m, plus the minimum accumulated distance to the grid point $(n-1, q)$ where q is the set of m values such that a path exists between $(n-1, q)$ and (n, m). Fig. 9 shows an example in which there are only three valid paths to any grid point (n, m), i.e., from $(n-1, m)$, $(n-1, m-1)$, and $(n-1, m-2)$. It further shows a "nonlinear" constraint that if the best path to grid point $(n-1, m)$ came from grid point $(n-2, m)$, then no path can lead from $(n-1, m)$. Formally, we can express such local continuity constraints on the path (for the example of Fig. 9) as

$$w(n) - w(n-1) = 0, 1, 2 \quad \text{if } w(n-1) \neq w(n-2) \tag{34a}$$

$$= 1, 2 \quad \text{if } w(n-1) = w(n-2) \tag{34b}$$

giving the modified form of (33) as

$$D_A(n,m) = d(T_n, R_m) + \min [D_A(n-1, m) g(n-1, m), D_A(n-1, m-1), D_A(n-1, m-2)] \tag{35}$$

where

$$g(n-1, m) = 1 \quad \text{if } w(n-1) \neq w(n-2) \tag{36a}$$

$$= \infty \quad \text{if } w(n-1) = w(n-2). \tag{36b}$$

The iteration of (35) is carried out over all valid m, for each value of n sequentially from $n = 1$ to $n = NT$, and the final desired solution is given as

$$D^* = D_A(NT, NR). \tag{37}$$

The optimum warping path $w(n)$ is determined by backtracking from the "end of the path" back to the beginning. For most word-recognition applications, the warping path need not be computed, only the accumulated distance D^* is required.

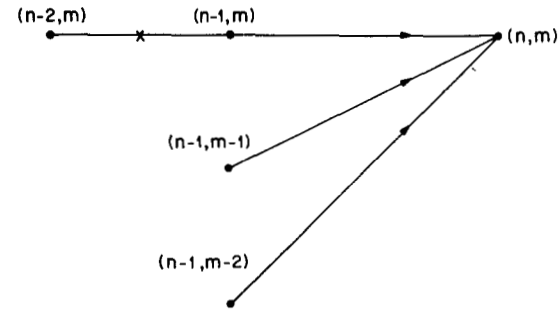

Fig. 9. One set of possible transitions to the grid point (n, m).

To illustrate a typical DTW warping region, Fig. 10 shows a grid size NT by NR. If the endpoint conditions of (11), and the local path constraints of (34) are used, the region in which the optimal warping path is required to lie is the shaded region in Fig. 10, i.e., a parallelogram bounded by lines of slope 2 and slope ½ from the grid extremal points (1, 1) and (NT, NR). The slope constraint of ½ is determined by the condition that the optimal path cannot be flat ($w(n) - w(n-1) = 0$) for two consecutive frames, and the slope constraint of 2 is determined by the condition that no path to the grid point (n, m) can come from any grid point lower than $(n-1, m-2)$.

This simplified discussion has shown that in order to implement a DTW algorithm, several factors must be specified including:

1) endpoint constraints on the path;

2) local path continuity constraints, i.e., the possible types of motion (e.g., directions, slopes) of the path

3) global path constraints, i.e., the limitations on where the path can fall in the (n, m) plane

4) distance measure.

In addition, there are differences in implementation of a DTW algorithm depending on whether the reference or test pattern is along the independent-time axis. Both Sakoe and Chiba, and Myers *et al.* [33], [34] have studied the effects of varying the above factors on both speed and performance of the DTW algorithm in actual speech-recognition systems. They have found that only small differences are found in performance for a farily wide range in variation of DTW parameters.

One of the most interesting results that came out of the experiments by Myers *et al.* [34] in working with different variants of the DTW algorithm is illustrated in Fig. 11. This figure shows the size of the parallelogram in which the optimal path can lie for three different ratios of reference length NR to test length NT, namely, 1 to 2, 1 to 3/2, and 1 to 1. It can be seen that when $NT = (3/2)NR$, the size of the parallelogram shrinks considerably (i.e., the region of the optimal warping path is much more constrained), and when $NT = 2NR$, only a single warping path is valid, namely a linear expansion of the reference pattern by 2 to 1. As such all the advantages of DTW time alignment are wasted, the more incommensurate the lengths of the test and reference pattern. Myers *et al.*, realizing this result, proposed a scheme in which *both* reference and test patterns are *linearly* warped to a fixed standard length, prior to DTW alignment. This scheme is illustrated in Fig. 12. In this manner, the path region is maximized and the DTW algorithm

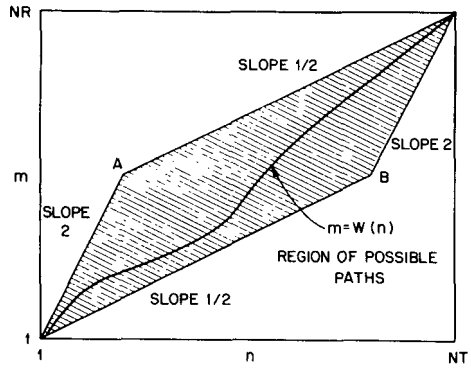

Fig. 10. Typical range for dynamic warping path with slope constraints of 2 to 1 and ½ to 1.

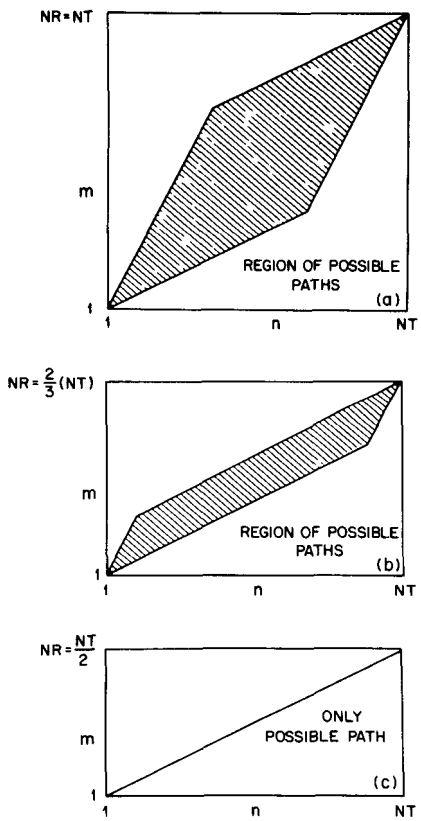

Fig. 11. The effects of relative duration of reference and test patterns on the range of the dynamic path.

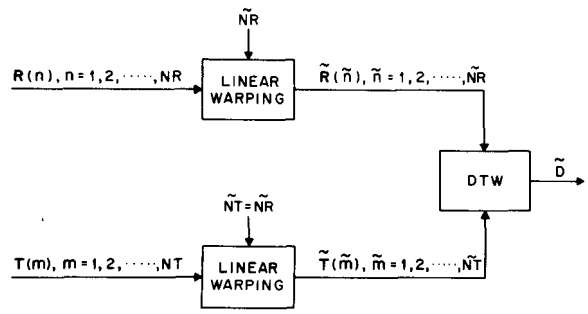

Fig. 12. A normalize-and-warp procedure for time alignment and distance computation.

has the best chance of matching the two patterns. Experimentation with this approach showed recognition performance comparable to or better than all other methods of dynamic time-warping alignment [34].

3) Endpoint Variants of DTW Algorithms: One of the major drawbacks in the simplified DTW algorithm presented in the preceding section is the assumption that the reference and test patterns must line up precisely at the initial and final frames. When reasonably accurate determination of the beginning and ending points of both patterns has been made, this constraint is acceptable and does not harm overall performance of the recognizer. However in cases where accurate endpoint detection cannot be made, the constrained endpoint DTW algorithm is inadequate. As such, several variations on the endpoint constraints have been proposed [35].

Fig. 13 illustrates three versions on the simple DTW algorithm discussed earlier. The first variant, called the constrained endpoints, 2 to 1 slope range (CE2-1) method is the one already discussed in which it is assumed that perfect alignment of endpoints of test and reference patterns exists. The second variant, called the unconstrained endpoints 2 to 1 slope range (UE2-1), retains all local path constraints, but relaxes the endpoint conditions to the set

$$1 \leqslant w(1) \leqslant 1 + \delta \tag{38a}$$

$$NR - \delta \leqslant w(NT) \leqslant NR \tag{38b}$$

where δ is an "offset" parameter of the DTW algorithm. Clearly, if $\delta = 0$, the UE2-1 DTW algorithm becomes identical to the CE2-1 method. Nonzero values of δ, however, increase the region of the (n, m) plane in which the path can lie, often significantly, as shown in Fig. 13(b).

The third variant of the DTW algorithm, called the unconstrained endpoints, local minimum (UELM) method, used the first relaxed endpoint constraint of (38a), but added tightened path constraints of the form

$$m^*(n-1) - \epsilon \leqslant m(n) \leqslant m^*(n-1) + \epsilon \tag{39}$$

where $m(n)$ is the range on m for searching for the optimum path for each value of n, $m^*(n-1)$ is the m index where $D_A(n-1, m)$ was minimum, i.e.

$$m^*(n-1) = \underset{m}{\operatorname{argmin}} [D_A(n-1, m)] \tag{40}$$

and ϵ is a range width parameter. As shown in Fig. 13(c), the UELM algorithm tracks the *locally* optimum path in order to estimate the globally optimum path with reduced computation. Clearly the final endpoint constraint is eliminated since the path itself determines the final endpoint alignment. This variant of the DTW algorithm is useful in recognition applications where only one of the endpoints is even approximately known, e.g., word spotting, or connected word recognition.

4) Some General Comments on DTW Algorithms: It is generally agreed that the introduction of dynamic time warping as a method of registering two speech patterns is one of the major breakthroughs that have made speech recognition practical for a wide range of conditions. The importance of

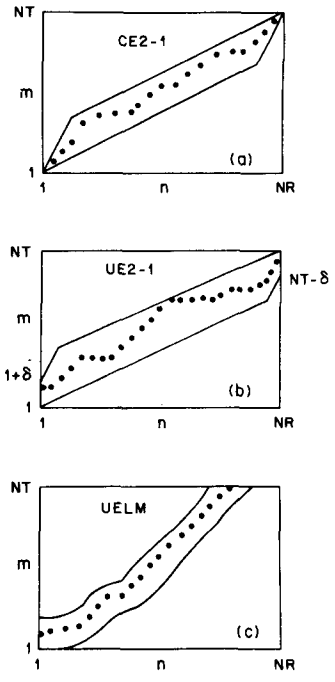

Fig. 13. Several variants on the simple DTW algorithm.

this class of methods has been manifest in a variety of areas including communications in which the classic algorithm of Viterbi for decoding convolutional codes has been shown to be a variant of a dynamic programming procedure [36], [37]. In the area of speech recognition, the importance of dynamic time warping was shown clearly by White and Neely [26] in experiments on isolated work recognition. These researchers showed that for polysyllabic words, distinct improvements in recognition performance were obtained over simple linear time alignment. We will see in later sections how dynamic time warping is a major component in systems for connected word recognition, as well as in syntactic processing for sentence recognition.

C. The Decision Rule for Recognition

The last major step in the pattern-recognition model of Fig. 1 is the decision rule which chooses which (reference) pattern (or patterns) most closely match the unknown test pattern. Although a variety of approaches are applicable here, only two decision rules have been used in most practical systems, namely, the nearest neighbor rule (NN rule) and the K-nearest neighbor rule (KNN rule).

The NN rule operates as follows. Assume we have V reference patterns, R^i, $i = 1, 2, \cdots V$, and for each pattern we obtain the average distance score D^i from the DTW algorithm. Then the NN rule is simply

$$i^* = \underset{i}{\text{argmin}}[D^i] \tag{41}$$

i.e., choose the pattern, R^{i^*} with smallest average distance as the recognized pattern. In some applications, as we will see later, explicit choice of i^* is not required; instead an ordered (by distance) list of recognition candidates is used. In this case, the set of distances D^i is reordered to give a new set $D^{[i]}$ such that

$$D^{[1]} \leqslant D^{[2]} \leqslant \cdots \leqslant D^{[V]} \tag{42}$$

and the set of indices that gave the new ordering $[i]$ is retained, i.e.

$$[i] = \text{Table}\ (i) \tag{43}$$

where Table (i) is the original index of the ith element in the reordered distance array.

The KNN rule is applied when each reference entity (e.g., word) is represented by two or more reference patterns, e.g., as would be used to make the reference patterns independent of the talker. Thus if we assume there are P reference patterns for each of V reference words, and we denote the jth occurrence of the ith pattern as $R^{i,j}$, $1 \leqslant i \leqslant V$, $1 \leqslant j \leqslant P$, then if we denote the DTW distance for the jth occurrence of the ith pattern as $D^{i,j}$, and if we reorder the P distances of the ith word so that

$$D^{i,[1]} \leqslant D^{i,[2]} \leqslant \cdots \leqslant D^{i,[P]} \tag{44}$$

then for the KNN rule we compute the average distance(radius)

$$r^i = \frac{1}{K}\sum_{k=1}^{K} D^{i,[k]} \tag{45}$$

and we choose the index of the "recognized" pattern as

$$i^* = \underset{i}{\text{argmin}}\ r^i \ . \tag{46}$$

Similarly to the NN rule, we can compute an ordered list of averaged distances (r^i) for cases when a list of recognition candidates is required.

The importance of the KNN rule is seen when P is from 6 to 12, in which case it has been shown that a real statistical advantage is obtained using the KNN rule (with $K = 2$ or 3) over the NN rule [16].

D. The Overall Word-Recognition System

At this point, we are ready to look in more detail at the operation of a particular isolated word-recognition system. The system we will use is the one which we have studied at Bell Laboratories for a number of years [12], [13], [15], [16]. Fig. 14 shows a block diagram of the recognition system based on LPC features, and using a DTW algorithm with the log likelihood (LPC) distance measure. A careful examination of this, or any other, isolated word recognizer, shows that the system has three distinct modes of operation, namely:

1) training, i.e., the acquisition of feature sets for each word in the vocabulary.

2) clustering, i.e., the creation of word reference templates from the training feature sets.

3) testing, i.e., the recognition of an unknown pattern by comparison with each reference pattern.

It is worthwhile examining what goes on in each of these

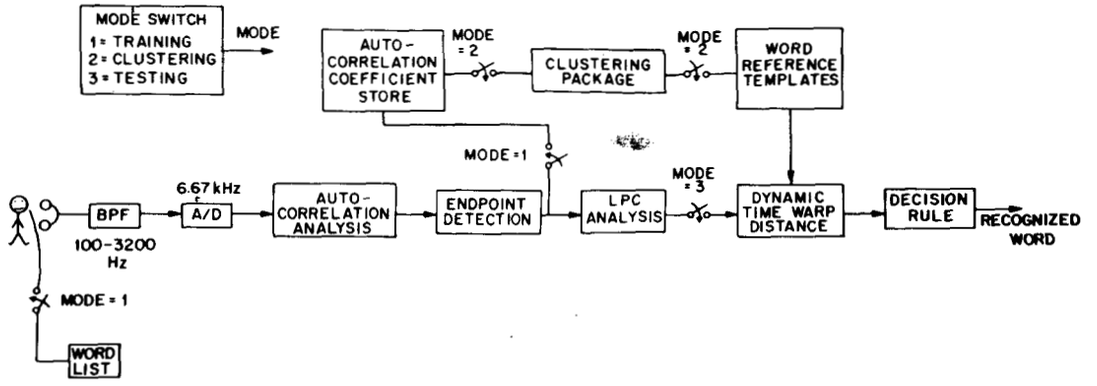

Fig. 14. Overall block diagram of LPC based isolated word-recognition system.

three modes more closely, as this will help explain how the recognizer actually works.

1) Training Mode: In the training mode, each speaker (for a speaker-trained system) or a set of speakers (for a speaker-independent system) recites each word of the desired vocabulary (one or more times) over some transmission system. For most of our own applications, the transducer used is the carbon button of standard telephone, and the transmission system is the standard telephone system for a dialed-up line (generally a local PBX). For commercial applications which do not require telephone transmission, generally a high-quality, close-talking, noise-canceling microphone is used to reduce the effects of the background environment (i.e., high-noise levels, extraneous conversation) on the speech quality.

The analog front end of the system of Fig. 14 consists of a standard bandpass filter from 100 to 3200 Hz (24 dB/octave attenuation skirts), followed by analog-to-digital (A/D) conversion at a 6.67 kHz rate (using a 15-bit convertor). From this point on, all processing is done digitally.

The next step in the processing is feature measurement in which a set of a autocorrelation coefficients are estimated every 15 ms (100 samples) using overlapping frames of 45 ms (300 samples) of speech which has been preemphasized and windowed.

At this point the process called endpoint detection must be carried out. Endpoint detection means literally finding the spoken word in the designated recording interval, i.e., separating the speech from the background sounds. This step is a crucial one in the recognizer for two reasions, namely:

1) Errors in endpoint location increase the probability of making recognition errors. Gross errors in endpoint location make reliable recognition impossible.

2) Proper lcoation of endpoints keeps the overall computation of the system to a minimum.

For reasonably quiescent recording conditions (i.e., a quiet room) endpoint location is a very simple procedure. However as the recording conditions degrade, the difficulty of endpoint location increases.

The technique used in the system of Fig. 14 is to use the contour (time pattern) of the zeroth autocorrelation coefficient (the signal energy) to locate endpoints by defining adaptive level and duration thresholds, and setting endpoints in terms of the energy contour exceeding the thresholds. Fig. 15 shows three examples of endpoints determined by the system for isolated spoken words. The endpoints of the word are denoted by the heavy vertical lines.

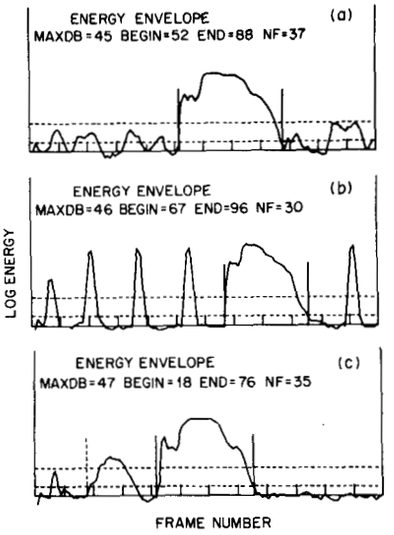

Fig. 15. Three cases of endpoint detection using energy contours.

In each of these examples the spoken word occurred in a background of conversation [Fig. 15(a)], line clicks [Fig. 15(b)], or extraneous lip noises [Fig. 15(c)] [38]. In all three cases, correct location of endpoints was made.

Following endpoint location the sets of autocorrelation coefficients within the spoken word are stored for use in the clustering mode. Thus, the training consists of iterative speaking and analysis of the vocabulary words, and storing the resulting feature sets in an appropriate place.

2) Clustering Mode: In the clustering mode, a conversion is made from isolated occurrences of feature sets for a word, to reference patterns to be used in the recognizer. There are three different methods which have been used to perform this conversion, namely:

1) direct conversion or casual training, in which a reference template is created for each occurrence of a feature set. Thus, if a speaker said each of vocabulary words two times during training, a total of $2V$ word templates are created. This method is used primarily in simple, speaker-trained systems where it is assumed that one or two spoken versions of each word are adequate for recognition.

2) Averaging conversion in which all the occurrences of a given word are averaged together (after some form of time alignment) to give a single reference template. This method provides a statistical gain over direct conversion since spurious recordings are downgraded by the averaging, if enough recordings of each word are made. For one commercial system ten recordings of each word are used for averaging [6].

3) Clustering conversion in which it is assumed that there are P occurrences of each vocabulary word, and they are grouped together to form Q clusters. Within each cluster the tokens (elements of the clustering analysis) have the property that they are "similar" (i.e., small distance to each other), and between clusters, the tokens have the property that they are dissimilar. For each such cluster, a single-word reference template is created using an averaging technique of the type discussed above. Clearly, the clustering analysis is most appropriate for obtaining speaker-independent templates; however it has been equally well applied to speaker-trained systems [39].

Fig. 16 illustrates the concepts of clustering for a set of 17 two-dimensional tokens (this is an articifial set). It can be seen that 15 of the 17 tokens fall into one of the three clusters labeled C1, C2, and C3 in Fig. 16. Each of these clusters would be represented by a single reference template. However it is also seen that two of the tokens (labeled A and B) are outliers, i.e., they are not close to any of the clusters. These outliers, if valid tokens (i.e., without recording artifacts) form single-element clusters are are individually represented as a template.

3) Testing Mode: The third mode of the recognition system proceeds initially as the training mode, i.e., a word is spoken, a set of features is measured, and the endpoint locations of the word are found. Following endpoint detection a full LPC analysis is performed on each frame of the word to give a test pattern $T(n), n = 1, 2, \cdots NT$ to be used in the DTW algorithm. This test pattern is optimally time aligned (using the normalize and warp procedure described above) with each of the V reference patterns, giving a distance score D_i, $i = 1, 2, \cdots, V$. The decision rule orders the distance scores and provides a best candidate or set of recognition candidates based on either the NN or KNN rules.

4) Illustration of the Recognizer Output: To set ideas firmly, Fig. 17 shows a plot of the minimum DTW accumulated distance function $D_A{}^*(n)$ versus n for each word in the vocabulary, where

$$D_A{}^*(n) = \min_m [D_A(n, m)]. \tag{47}$$

In this example, the spoken word was the letter /Q/ and the vocabulary consisted of the letters of the alphabet. A total of two templates were used for each vocabulary word. By looking at $D_A{}^*(NT)$ (i.e., at the right-hand edge of the plots), it can be seen that the two /Q/ templates achieved the smallest distance scores, and that the two /U/ templates achieved the next smallest scores. No other letter achieved an acceptably small distance to be considered. It can also be seen that from frame 10 to the end, the rate of accumulation of distance for /Q/ and /U/ were about the same, as might be expected phonetically.

Also shown in Fig. 17 are two recognition features that serve to reduce computation, and increase flexibility of the

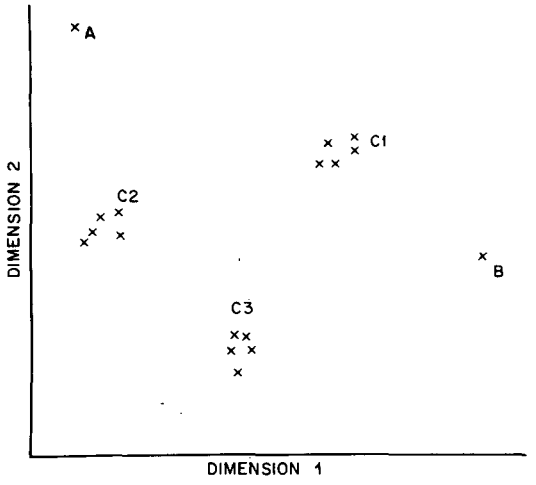

Fig. 16. Illustration of clustering of tokens in a two-dimensional space.

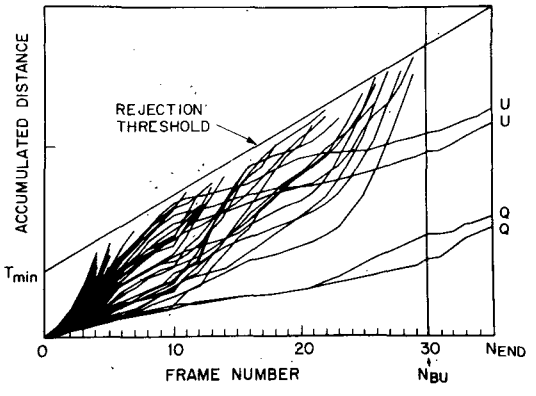

Fig. 17. Plots of accumulated distance versus frame number for a 26-word vocabulary and for the spoken test word /Q/.

system. The first, called the rejection threshold, is a curve of accumulated distance which bounds the DTW search. Thus, if the minimum accumulated distance $D_A{}^*(n)$ at frame n exceeds the threshold $\hat{T}(n)$, then the DTW search is terminated and the reference template is given an infinite distance. As shown in Fig. 17, $\hat{T}(n)$ is generally of the form

$$\hat{T}(n) = (T_{min} + (n-1)T_{slope}) \tag{48}$$

where T_{min} and T_{slope} are parameters of the distance function. (For our system $T_{min} = 3.0$, $T_{slope} = 0.7$.)

The second extra recognition feature is the backup frame, labeled N_{BU} in Fig. 17. This is essentially an alternative word ending frame based on the assumption that a breath noise was made at the end of the word and included within the word interval. The backup frame is computed directly from the word energy contour [16], and is used as an early "stopping" frame in the DTW algorithm.

5) Some General Comments on the Word Recognizer: The system of Fig. 14 has several desirable features that have led to its use in a variety of applications, as will be discussed in Section III. These include the following.

1) It can be used as either a speaker-trained, or a speaker-independent system with no modifications.

2) It can be used with any word vocabulary.

3) It is modular in its three main steps, and alternative algorithms (i.e., new feature sets, DTW methods, etc.) can be readily used and tested.

4) It is an all-digital implementation.

The advantages have literally made the recognizer a single module in a number of larger task-oriented systems.

6) Results on Isolated Word Recognition Tests: The isolated word recognizer of Fig. 14 has been used in a wide variety of evaluation tests, and the results of these evaluations are given in Table I. This table gives overall recognition accuracy for different vocabularies and training modes. (The speaker mode column designates whether the system was used as a speaker-trained recognizer or a speaker-independent recognizer. It also designates the procedure used to obtain templates, where appropriate.) A quick glance at the table shows that recognition accuracy varies from 79 to 100 percent, and more importantly, it is seen that the accuracy is not a function of the vocabulary size, so much as it is a function of vocabulary complexity. Hence a small size vocabulary consisting of many similar sounding words (e.g., A, J, K, or B, D, E, G, P, T, V, Z, etc.) is considerably more complex than a vocabulary of 200 polysyllabic, distinctly different words (e.g., the Japanese cities list).

The results given in Table I are chronologically ordered. Hence, when the same vocabulary is tested with an improved system (e.g., 54-word vocabulary using clustered speaker-independent templates), the recognition accuracy will often show substantial improvement. A general result seen from the data in Table I is that high-recognition accuracy (>95 percent) can be obtained in a speaker-trained system by careful creation of word reference templates (even for highly complex vocabularies), and for a speaker independent system when noncomplex vocabularies are used.

More insight into the performance of the isolated word recognizer (operating in a speaker-independent mode) can be gained by examining the results presented in Fig. 18. This figure shows plots of recognition accuracy as a function of the number of templates per word used in the recognizer, the value of K for the KNN rule, the word candidate position, and the method used to obtain the word templates. Parts (a) and (b) of this figure are results for templates obtained from a sophisticated clustering procedure, and parts (c) and (d) are similar plots for templates obtained by random selection from the tokens used in the clustering. Parts (a) and (c) are results for the top recognition candidate, and parts (b) and (d) are results for the top five choices. If we first examine Fig. 18(a), we see the following.

1) For the clustered template set, recognition accuracy steadily increases as the number of templates per word increases from two to about eight, at which point no real increases in recognition accuracy are obtained.

2) For small numbers of templates per word, the KNN rule performs best with $K = 1$; for large numbers of templates per word, the KNN rule with $K = 2$ produces the highest accuracy.

For the vocabulary used in making these plots (i.e, the digits, the letters of the alphabet, and three command words), recognition accuracy goes from about 62 percent (one template per word), to about 79 percent (12 templates per word) for the top candidate.

TABLE I
RECOGNITION SCORES FOR SEVERAL ISOLATED WORD-RECOGNITION SYSTEMS

Reference	Vocabulary	Speaker Mode	Accuracy
Itakura [12]	200 Japanese cities	Trained - 1 Talker	97.3%
Itakura [12]	A-Z, 0-9	Trained - 1 Talker	88.6%
Rosenberg [13]	84 words, cities, numbers days, airports	Trained - 10 Talkers	91.6%
Rosenberg [45]	A-Z	Trained - 10 Talkers	79.5%
Rabiner [15]	54 Computer words	Independent - Pseudo-clustered, 8 Talkers	85.4%
Rabiner et. al. [16]	A-Z, 0-9, STOP ERROR, REPEAT	Independent - Clustered 28 Talkers	79%
Rabiner et. al. [16]	0-9	Independent - Clustered 110 Talkers	98.2%
Rabiner, Wilpon [40]	54 Computer words	Independent - Clustered 40 Talkers	96.5%
Rabiner, Wilpon [39]	A-Z, 0-9, STOP, ERROR, REPEAT	Trained-Clustered 3 Talkers	97%
Rabiner et. al. [41]	20 Names, 10 Digits, 7 Command Words	Trained - 6 Talkers	100%
Rabiner, Wilpon [42]	561 Words and Phrases	Trained - 2 Talkers	95%

Fig. 18(b) shows similar trends in accuracy scores for the top five recognition candidates with overall accuracy going from 88 to 98 percent as a function of the number of templates per word.

For the case of randomly chosen templates [Figs. 18(c) and (d)], it is seen that the same trends exist in the data. However, the absolute recognition scores are from 4 to 12 percent lower, conclusively demonstrating the performance improvements obtained from using clustered templates.

Based on the results given in Fig. 18, and those obtained in subsequent tests, the following general conclusions have been drawn.

1) Clustering methods are effective for finding structure in a group of tokens representing multiple occurrences of an isolated word, and can be used to provide a robust, accurate set of speaker-independent word-reference templates.

2) Performance obtained from clustered tokens is significantly better than performance obtained from randomly chosen tokens.

3) The KNN rule provides higher recognition accuracies for values of K of 2 and 3 than for $K = 1$ (NN rule) for a large number of templates per word.

4) Six to twelve templates per word are adequate for representing a large population of talkers (on the order of 100-1000).

These results have been used in the speaker-independent, isolated-word recognizer at Bell Laboratories, and the performance of the system has not changed in a three-year period,

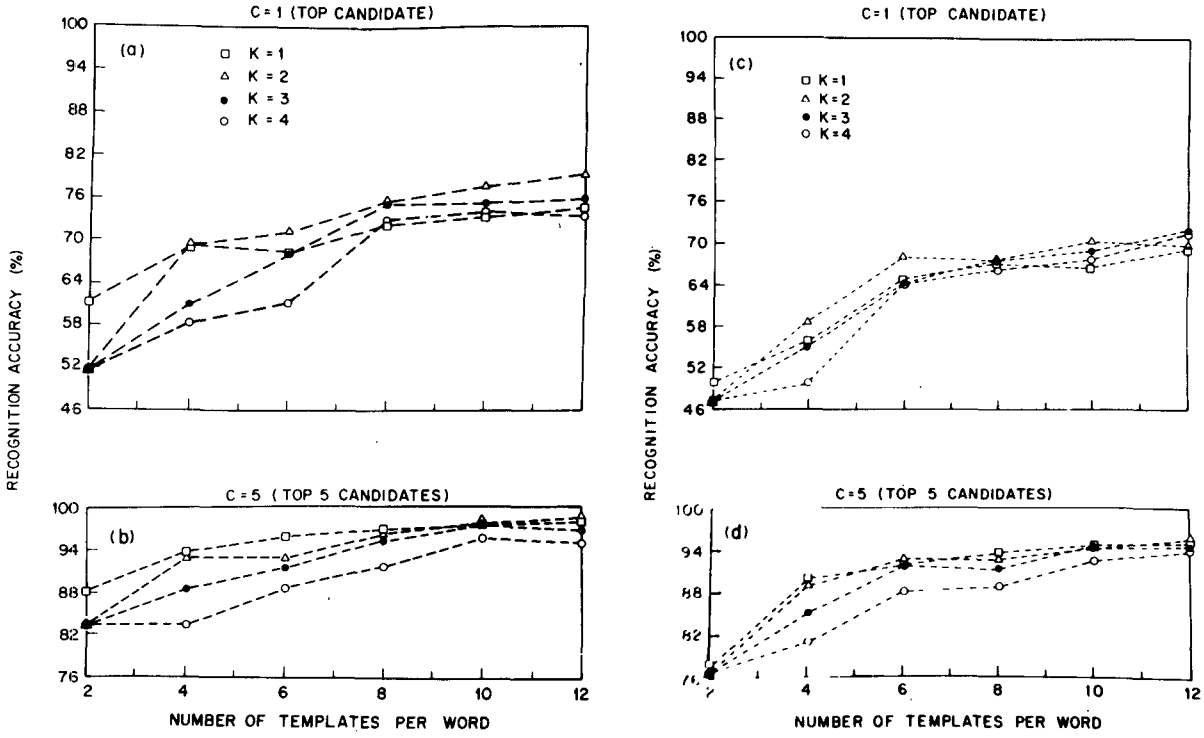

Fig. 18. Recognition accuracy for speaker-independent systems as a function of several recognition-system parameters.

even though the word templates were created at the beginning of the test period.

7) New Directions For Isolated Word-Recognition Systems: Although it might seem, from the discussion of the previous sections, that it has all been done, there remains several open issues in isolated word recognition that are the topics of active research. One such issue is the question of how to improve the performance of the system for complex vocabularies containing many similar sounding words. Although there is inherently no "perfect" solution to this problem, recent work by Rabiner and Wilpon [43] suggests that a two-pass approach to recognition can make improvements in the recognition accuracy. The way in which such a system would work is as follows. The vocabulary words are first grouped into "equivalence classes" based on phonetic and acoustic similarity. Using the equivalence class representation of the vocabulary, the first recognition pass would determine the equivalence class in which the unknown word occurred. The second pass of the recognizer would then use an optimally determined distance weighting curve to distinguish between "similar" words, i.e., words within the acoustic equivalence class. Preliminary results using such a system on the vocabulary of letters and digits indicates improvements in recognition accuracy of 3 to 8 percent.

Another place in which improvements can be made is in the endpoint location method. Errors in locating endpoints account for a large percentage of the errors made in most recognizers. Unfortunately, we have no panacea to this universal problem.

Finally, it is incumbent upon us to make some comments about practical (hardware) implementations of isolated word recognition systems. Commerically available systems cost from \$100 to \$80 000 per channel. The more sophisticated, higher performance systems (i.e., the type we have been discussing here) range from about \$2000 to \$80 000. With the advent of increased digital processing power of modern microprocessors, it is estimated that the costs of the high-performance systems will tumble to the \$100 range in the foreseeable future (five-year time period). The system which we have discussed in this paper is currently being implemented in digital hardware [44] using an NMOS microprocessor (Intel 8086) in conjunction with a specially designed digital-speech processor based on the TRW 1010J high-speed multiplier-accumulator chip. A block diagram of the hardware configuration is given in Fig. 19. For a 40-word vocabulary (speaker-trained) a recognition time of about ¼ s is required. The parts cost of the system is about \$1800 (including memory for 160 templates), and it is anticipated that such a system would cost about \$300 wihin three years. Thus, it is felt that the high-performance word-recognition systems will become more ubiquitous in the foreseeable future as the cost falls sufficiently to justify its application.

8) Summary of Issues in Isolated Word Recognition: If we review the ideas we have been discussing in this section, we see that there are a number of factors that must be specified for the implementation of an isolated word recognition system. These factors include the following.

1) Speaker-trained or speaker-independent system. We have seen that this affects the training and the decision rule used.

2) Vocabulary complexity. This factor affects overall recognition accuracy.

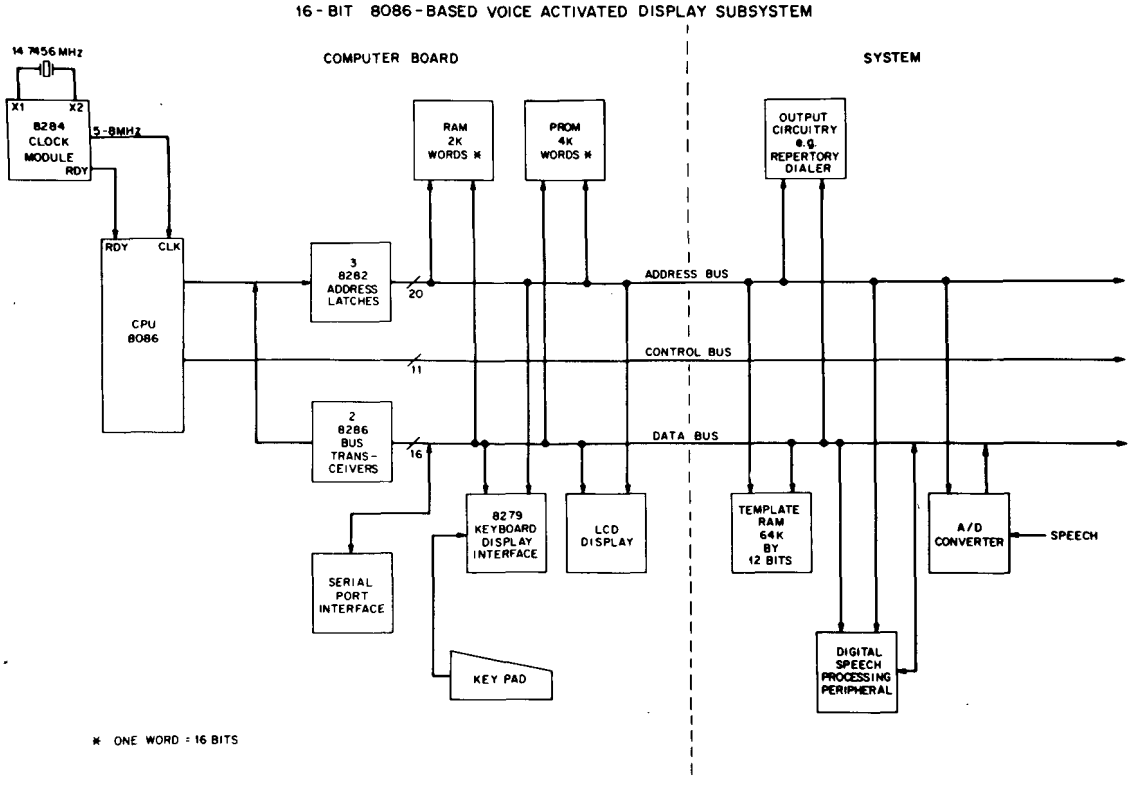

Fig. 19. Block diagram of a hardware structure for implementing the LPC-based isolated-word recognizer.

3) Transmission system (e.g., high-quality microphone, close talking, noise-cancelling microphone, telephone). This factor affects the endpoint location algorithm.

4) System complexity, e.g., the amount of computation for recognition. This factor affects the system response time.

Each of these factors must be taken into consideration for proper design of the isolated word recognition system.

III. TASK-ORIENTED APPLICATIONS OF ISOLATED WORD RECOGNITION

Although we have shown that isolated word-recognition systems will often perform adequately, the power of speech recognition lies in its ability to perform a given task reliably, i.e., when the isolated recognizer is only a single component in a larger, more general system. It will be shown in both this section, and in the following one, that the power of the isolated word recognizer is increased substantially when embedded in a higher level task. In this section, we will discuss three fairly simple task-oriented applications, namely:

1) a voice controlled repertory dialer system [41];
2) a directory listing retrieval system [45], [46];
3) a connected digit recognizer [8], [18], [19].

The task in each of these systems is specified by a simple set of rules which essentially limits the possible set of recognition outputs. Each of the sets of rules is specified as a table of possible recognition sequences at each point in the system. In Section IV, we will present a more theoretical approach to the idea of using task constraints to increase reliability and improve recognition performance.

A. Voice-Controlled Repertory Dialer System

A voice-controlled repertory dialer system is a speech recognizer which is capable of responding to single-word commands, and which can perform the following tasks:

1) build an inventory (a repertory) of spoken names for which the telephone number is determined (from voice input);

2) access any name from the repertory and have the associated telephone number dialed directly;

3) automatically dial any "all-digit" telephone number spoken into the system;

4) edit the repertory of spoken names to add names (and telephone numbers), delete names, and modify telephone numbers.

Thus, the voice controlled repertory dialer is essentially equivalent to a push-button repertory dialer with the button pushing being repalced by speech commands.

Fig. 20 shows a block diagram of the dialer system. The nominal vocabulary for using the system is the 17-word vocabulary consisting of

1) digits: 0-9

2) command words: Offhook, Hangup, Add, Delete, Modify, Error, Stop.

In addition the vocabulary for each user contains the set of names appearing in the repertory. This system could be classified a hybrid recognizer since the 17 command words could be recognized using speaker-independent templates, and the

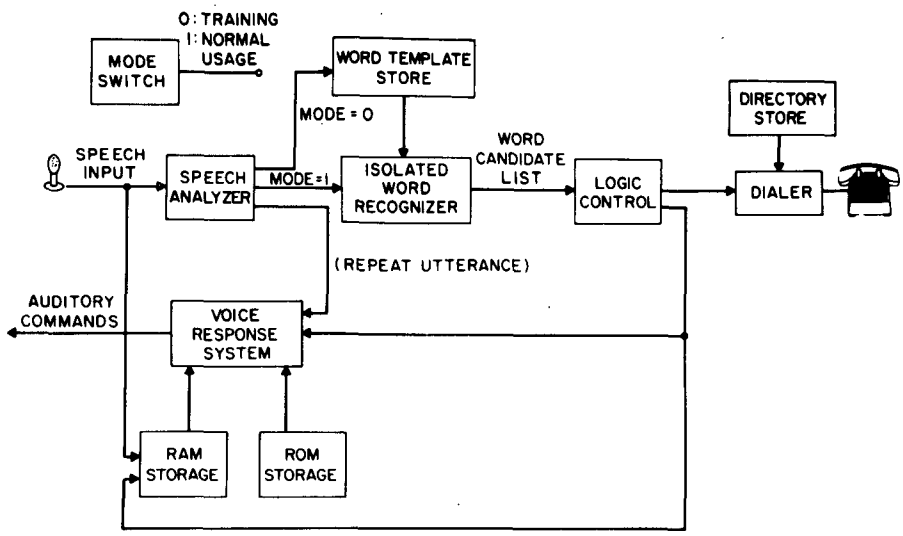

Fig. 20. Block diagram of the voice controlled repertory dialer system.

repertory names would be recognized using speaker-trained templates.

As seen in Fig. 20, the major components of the repertory dialer system are

1) real-time speech analyzer which measures LPC features (in real time) for the word recognizer;

2) an isolated word recognizer to determine the spoken word;

3) a voice-response system to guide the user as to when to speak, and to provide feedback as to the word (or words) recognized;

4) logic control to guide the operation of the system;

5) a dialer to outpulse the desired telephone number.

There are four major storage memories that are required in the system, namely:

1) a word template store to store the reference patterns;

2) a directory store to store telephone numbers associated with each directory name;

3) RAM voice response storage to store coded versions of the acoustic waveform of each name in the repertory;

4) ROM storage to store standard voice-response commands for prompting and feedback to the user.

The way in which the dialer operates is as follows. Initially the user must create his repertory. He does this by speaking the "Add" command, following a system prompt to speak (a double beep). The system asks the user to enter the new repertory name, and the acoustic waveform is stored (in coded format) in the RAM memory of the voice-response system. The system then obtains a recognition template for the name as well as the telephone number (spoken as a set of isolated digits), and stores this information in the template and directory stores, respectively. This process continues until all repertory names are entered. By using the "Delete" or "Modify" commands, names can be deleted from the repertory, or telephone numbers can be modified. These three editing commands (namely, Add, Delete, Modify) can be invoked at any time when the system is running.

Following training, the system can be used as a repertory dialer by speaking the command "Offhook" following a signal prompt. This command (acknowledged to the user by a single beep prompt) takes the telephone off the hook so a number can be dialed. Following the single beep the user can speak any previously recorded repertory name, or speak a sequence of isolated digits. The system verifies the recognition via the voice-response system and waits for an "Error" response. If none is recognized, the telephone number is dialed. If an error response is recognized, a request to repeat the spoken sequence is made, and the process continues.

There are several points worth noting about the voice dialer. The first is that all communication between the user and the system is by voice. No visual display of any type is needed to train or to use the system. The voice response commands (as stored in ROM memory) include the 13 phrases shown in Table II, and the 7 command words, and 10 digits of the vocabulary. RAM memory space has to be allocated for each of the names in the repertory—typically 10 to 20 names should be sufficient. If all voice response commands are coded to 24 kbits/s [47], [48], using a waveform coding technique like ADPCM, a total of about 720 000 bits (30 s × 24 000 bits/s) are required for ROM storage, and 360 000 bits (15 s × 24 000 bits/s) are required for RAM storage. With LPC coding, the storage requirements are reduced further by a factor of ten or more (although the coder/decoder costs increase substantially).

A second feature of the dialer is that the system responds only to isolated word inputs. Thus the user may hold a conversation while the dialer is operating, and the system will not be triggered unless an isolated version of one of the command words is recognized. In order for a word to be recognized, it must have a distance score within prescribed limits, and it must have a considerably smaller distance than the next likely recognition candidate. The likelihood of such events occurring during conversational speech is very small.

Another aspect of this system is that the vocabulary is partitioned for recognition into the following sets:

1) SET 1—7 command words
2) SET 2—L names, 10 digits, word STOP

TABLE II
PHRASES USED BY THE VOICE-RESPONSE SYSTEM OF THE REPERTORY DIALER

1. After each tone say the specified word.
2. Please repeat.
3. Please repeat the command.
4. Please repeat the number.
5. At the beep, speak the name to be added.
6. Please repeat the name to be added.
7. At the beep say the word (—).
8. Please enter phone number.
9. Please repeat the name to be deleted.
10. Please enter the name to be deleted.
11. Please enter new phone number.
12. Please verify.
13. Please repeat the name whose phone number is to be changed.
14. "Beep".

3) SET 3—10 digits
4) SET 4—STOP and ERROR.

Thus, in the worst case the recognizer must choose among $(L + 11)$ possible candidates. However, even for that case, more information is present in the task. If the recognizer finds a digit, the task knows that it must be part of a four-digit string. If no such string is found, the task can choose the best recognition candidate among the set of the names and the word STOP. Similarly, if a string of digits is spoken and the recognizer matches the first digit to a name, the task can correct the word to the most likely digit based on the recognition of subsequent digits.

To demonstrate the effectiveness of these simple task constraints on the recognition, the dialer system was tested on a vocabulary of 20 names (chosen from the Acoustics Research Department) and the 17 standard words. Six talkers (three male—three female) each spoke 170 full commands (a total of 770 words) over a two-week period. The commands were chosen to test the full range of the dialer system. The results of the test showed *no recognition errors for any talker.* This result conclusively demonstrates that a task-oriented recognizer can be implemented in a reliable manner if one can take advantage of some of the natural constraints of the task, the vocabulary, and the recognizer.

B. Directory Listing Retrieval System [45], [46]

Another interesting, task-oriented application of isolated word recognition is the problem of information retrieval from a directory. In particular we have studied the problem of retrieving a directory listing from a spoken spelled name. A

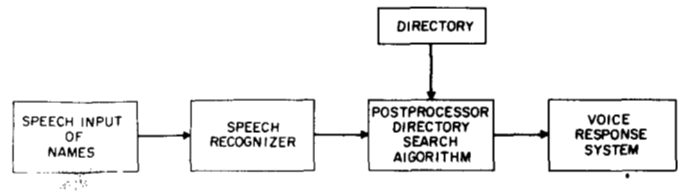

Fig. 21. Block diagram of directory listing retrieval system from speech input of names.

block diagram of a canonic directory listing retrieval system using speech input is shown in Fig. 21. The user provides to the system the required input information (in the form of spoken commands) which specifies an entry in the directory for which some information is desired. The directory search technique searches the directory to find the best match (in some sense) to the given input, and either provides directory information to the user (generally via a voice response system), or requests additional input information to help resolve ambiguity in the directory search.

In the system we have studied [45], [46], the input information is a sequence of isolated letters (for the last name) followed by the isolated command word /STOP/, followed by initials (whichever ones are known) and a final /STOP/ to denote the end of the string. Thus, a typical input would be of the form

L/E/E/STOP/K/STOP

for directory listing retrieval foı the name Kim Lee. In speaking the above string of words, each word has to be spoken in isolation (i.e., a distinct pause or gap of at least 50 ms between words is required). In the system which was studied, only initials are used for the first and middle names, and any number of initials (from 0 to 2) can be used. Up to six letters of the last name are used since, for the directory that was investigated, almost no new information about the name is obtained for more than six letters of the last name.

The speech recognizer treats each separate input as an isolated word, and does speaker independent recognition using a 27-word vocabulary (A-Z, STOP) with 12 clustered templates per word. To illustrate a typical recognition output, Table III shows the sets of ordered (by distance) recognition candidates for the spoken string

R/A/B/I/N/E/STOP/L/R/STOP.

Only the smallest 10 distances are shown for each input in the string. A circle is drawn around the correct candidate for each string position. It can be seen that, for the example, two recognition errors were made, namely the letter /A/ in position 2 was recognized as the ninth candidate, and the letter /N/ in position 5 was recognized as the second candidate. As will be shown shortly, it is the job of the postprocessor directory search algorithm to use the distances generated by the recognizer to find the correct name in the directory.

Thus, the output of the speech recognizer is a matrix of (unordered) distance scores $\tilde{D}$, where the $\tilde{D}_{ij}$ entry is the recognition distance between distance between vocabulary

TABLE III
TYPICAL EXAMPLE OF THE CANDIDATES AND DISTANCE SCORES FOR A SPOKEN NAME

	Candidates/Distances									
WORD # 1	(R) .456	STOP .577	I .605	M .646	Y .723	L .750	F .782	N .820	S .834	X .840
WORD # 2	J .391	G .409	C .428	K .430	Z .465	T .490	U .539	D .541	(A) .573	P .605
WORD # 3	(B) .318	D .323	E .330	G .343	P .355	T .360	U .429	Z .458	C .557	A .614
WORD # 4	(I) .338	Y .376	R .600	ST .673	F .788	M .796	S .829	L .872	U .872	X .880
WORD # 5	M .468	(N) .478	X .563	F .577	STOP .579	S .613	L .655	U .783	U .786	K .799
WORD # 6	(E) .203	P .287	B .299	T .316	D .320	G .331	U .453	Z .468	A .549	C .555
WORD # 7	(STOP) .291	X .573	M .588	F .596	S .610	L .646	N .665	I .710	C .725	U .731
WORD # 8	(L) .288	Q .644	F .675	STOP .686	U .699	O .721	M .837	W .890	X .925	S .931
WORD # 9	(R) .268	I .531	Y .598	ST .604	M .612	L .677	F .713	S .769	O .778	N .791
WORD # 10	(STOP) .254	X .543	M .580	L .581	F .584	S .596	I .677	N .684	U .703	Q .751

RABINER LR

LOCATION IS: MH
ROOM NUMBER: 2D533

word i and the jth spoken word in the input string. If we assume there are P words in the vocabulary ($P = 27$) and an L word string was spoken, then $\tilde{D}$ is of the form

$$\tilde{D} = \begin{bmatrix} \tilde{D}_{1,1} & \tilde{D}_{1,2} & \cdots & \tilde{D}_{1,L} \\ \tilde{D}_{2,1} & \tilde{D}_{2,2} & & \vdots \\ \vdots & \vdots & & \vdots \\ \tilde{D}_{P,1} & \tilde{D}_{P,2} & \cdots & \tilde{D}_{P,L} \end{bmatrix}. \tag{49}$$

The postprocessor search algorithm must find the string (or strings) in the directory which has minimum distance $D(S)$, where S is an L-component (letter) string defined as

$$S = (l_1, l_2, \cdots, l_L). \tag{50}$$

The distance of any string S can be computed from the matrix $\tilde{D}$ as

$$D(S) = D(l_1, l_2, \cdots, l_L) = \sum_{j=1}^{L} \tilde{D}_{l_j,j} \tag{51}$$

where we have used a natural correspondence between word numbers (l_j) and vocabulary words, i.e., letter A is word 1, letter B word 2, etc.

Based on (51), a string distance is defined for each string in the directory. The purpose of the search algorithm is to find the minimum-distance string with a small number of probes. The key to the search algorithm that is used is the assumption that letter confusions in the word recognizer occur primarily with known classes of acoustically similar letters, i.e., B and V are confused, A and K and J, etc. Thus, even if a spoken letter is not recognized correctly (i.e., it does not have minimum distance) the acoustic class to which it belongs is recognized correctly most of the time. This characteristic of the recognizer is exploited by reogranizing the directory in terms of acoustically similar letter classes (rathern than alphabetically) and then by using a searching strategy that exploits this new classification scheme. Hence the manner in which the correct name is generally found is similar to the procedure used by a grade schooler in looking up in a dictionary the spelling of a word which he does not know how to spell. A first guess at the approximate name (spelling) is taken from the distance matrix and all names that are acoustically equivalent (in a specified sense) to this first guess are accessed, their string distances are calculated using (51) and a record is kept of the best name candidates. A second guess at the approximate name is taken from the distance matrix (in a fairly simple way) and again all acoustically similar names are accessed and their distances are calculated. When the best distance in a new acoustic name class becomes too large, the procedure terminates and the best name (or names) are output to the user. The details of the search procedure are given in [46] and [49] and will not be discussed here. Instead, we will highlight some of the performance characteristics of the overall directory listing retrieval system.

The performance of the word recognizer is best expressed as a curve showing the error rate $E(i)$ as a function of the rank i in the ordered distance vector, i.e., the percentage of letters which are not in the top i positions of the ordered distance vector. Fig. 22 shows such a curve for three cases, namely, the average for 20 talkers used to test the system, the curve for the best talker, and the curve for the worst talker. It can be seen that for the average talker, the correct letter is not found in the first position about 30 percent of the time, and, in fact, is not found in the top seven positions about 3 percent of the time. Thus the probability of recognizing a name based on the top recognition candidate for each letter position is rather low—on the order of 0.1, and thus the search algorithm essentially provides the leverage required to obtain high name accuracy in spite of low-word-recognition accuracy.

At the output of the directory search procedure, one of three conditions can occur, namely the following.

1) A single string has been found whose distance is sufficiently small to accept this string as the spoken name.

2) No string has been found whose distance is acceptably small. In the case, a request is made to the user to repeat the spoken input.

3) More than one string has been found whose string distances are sufficiently small, i.e., there is ambiguity as to what is the correct choice among several candidate strings. The way in which we handle such cases will be discussed below.

Based on the above possibilities, Table IV(a) shows the performance characteristicsof the overall directory testing retrieval system for the cases when 0, 1, or 2 (when possible) initials were specified for the Bell Laboratories directory of 18 000

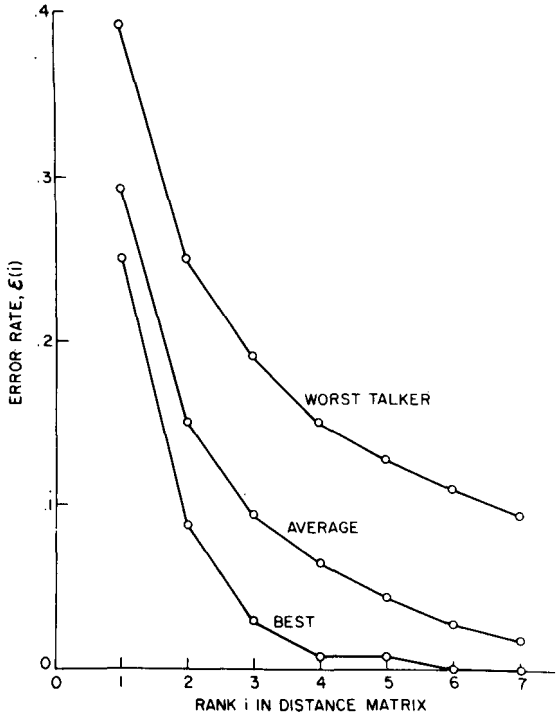

Fig. 22. Recognition error rate as a function of rank in the distance matrix for the worst, average, and best talkers.

TABLE IV
RESULTS ON THE DIRECTORY LISTING RETRIEVAL TASK. (a) ONE-PASS SEARCH. (b) TWO-PASS SEARCH.

NO INITIALS	CORRECT ANSWERS	ERRORS	UNRESOLVED
0	40.5	8.1	51.4
1	74.9	8.9	16.2
2	92.0	5.0	3.0

(a)

NO INITIALS	CORRECT	ERROR	UNRESOLVED
0	PASS 1: 34.9	1.5	---
	PASS 2: 92.4	2.6	5.0
1	PASS 1: 66.2	1.2	---
	PASS 2: 96.8	1.9	1.3
2	PASS 1: 87.8	0.8	---
	PASS 2: 98.3	1.1	0.6

(b)

names. It can be seen that for 0 initials, over half of the cases are unresolved, whereas for 2 initials specified, only 3 percent of the cases are unresolved. However, even for 2 initials, only 92 percent of the names are correctly determined, i.e., there is a 5 percent error rate. Such an error rate is too high for this system to be of any practical utility.

To improve the performance of the system, two modifications were made, as follows.

1) A distance threshold was defined such that two strings were defined as equivalent if the difference in their string distances was below this threshold.

2) When string ambiguity was found, a request for additional information (in this case the organization number at Bell Laboratories) was made and the recognizer was used a second time to recognize the new spoken input.

A flow diagram of this "two-pass" directory listing retrieval system is shown in Fig. 23. The acoustic recognizer and the directory search algorithm are seen to communicate via direct control paths, and from files in which all partial results are retained.

Table IV(b) illustrates the improvements in performance using the two-pass recognition-search strategy. For the case of 0 specified initials, the percentage of correct names jumps to 92.4 percent (after the second pass), whereas for two initials specified, a name accuracy of 98.3 percent with an error rate of 1.1 percent is attained. These improvements are truly significant, and again illustrate the leverage in improving recognition accuracy provided by a well-designed task.

C. Connected-Word Recognition Using Isolated Word-Reference Patterns

Interestingly, one of the most important applications of the techniques of isolated-word recognition has been the area of connected-word recognition. Before discussing this application, we must first define connected-word recognition, and distinguish it from continuous-speech recognition. In connected-word recognition, the spoken input is a sequence of words from a specified vocabulary, and the recognition is performed based on matching isolated word-reference templates. Typical examples include connected digit strings where the vocabulary is the set of 10 digits (0-9), or connected-letter recognition (e.g., for spelling names, words, etc.) where the vocabulary is the set of 26 letters (*A-Z*). Continuous-speech recognition, on the other hand, generally involves recognition from basic units of speech, e.g., phonemes, syllables, demisyllables, diphones, dyads, etc., and hence implies the need for some form of segmentation of the speech into the units, and labeling of the units. (There do exist speech recognizers (e.g., CMU's HARPY system) which combine aspects of both connected-word and continuous-speech recongition). It is generally acknowledged that continuous-speech recognition is a considerably more difficult problem than connected-word recognition. However, there are many applications where connected-word recognition serves a useful purpose.

A block diagram of a fairly general pattern-recognition structure for connected word recognition is given in Fig. 24. This block diagram is almost identical to the one for the isolated-word recognizer, with one major exception. This exception is the feedback loop shown at the output of the decision rule box which, at the end of each recognition (level), feeds back to the DTW algorithm a set of estimates of where in the test string the current local matches end. In this manner, the DTW algorithm can progressively build up a set of matches to the test string and, at the end of the search, determine an ordering of matches according to accumulated distance scores.

The remaining components of the connected-word recognizer, namely, the acoustic analyzer, the reference templates,

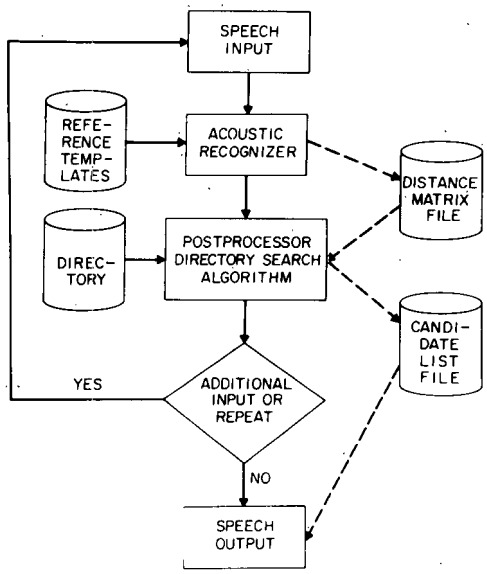

Fig. 23. Flow diagram of a two-pass directory listing retrieval system.

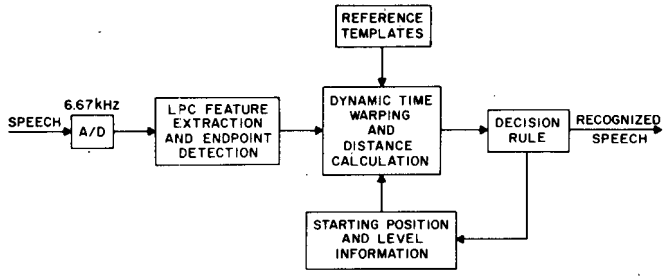

Fig. 24. Block diagram of a connected word-recognition system.

the DTW algorithm, and the decision rule are essentially the same as used for isolated-word recognition.[3] Thus, one key point about connected-word recognition is that the reference templates are isolated-word templates. The only other point of note is that the DTW algorithm cannot be a constrained endpoint algorithm, since no word boundaries are known *a priori*. Hence, a DTW algorithm such as the UELM algorithm, must be used in this application.

To understand how the connected-word recognition is solved, we must first define some terms. We denote the unknown test string pattern as $T(m)$, $m = 1, 2, \cdots, M$, where $T(m)$ represents a string (of unknown length) of words. The test string $T(m)$ is to be time registered with a sequence of L reference patterns $R_{q(1)}(n)$, $R_{q(2)}(n)$, $\cdots$, $R_{q(L)}(n)$, where each $R_{q(n)}$, $k = 1, 2, \cdots, L$ is one of a set of V reference patterns (the word vocabulary) R_v, $v = 1, 2, \cdots, V$. The length of the vth reference pattern is denoted as N_v. We define a "super" reference pattern [8], [50] $R_{q(1)q(2)^S \cdots q(L)}$ as the concatenation of the L reference patterns $R_{q(1)}, R_{q(2)}, \cdots, R_{q(L)}$, i.e.

$$R^S = R_{q(1)} \oplus R_{q(2)} \oplus \cdots \oplus R_{q(L)} \tag{52}$$

[3] The decision rule, as in the isolated-word recognizer, chooses the word string with minimum distance. If one considers the class of all possible time warps for a string (or for a word), then alternative decision rules could be considered.

or

$$R^S(n) \begin{cases} = R_{q(1)}(n - \phi(0)) & 1 + \phi(0) \leqslant n \leqslant \phi(1) \\ = R_{q(2)}(n - \phi(1)) & 1 + \phi(1) \leqslant n \leqslant \phi(2) \\ \vdots & \\ = R_{q(L)}(n - \phi(L-1)) & 1 + \phi(L-1) \leqslant n \leqslant \phi(L) \end{cases} \tag{53}$$

where the length function $\phi(l)$ is defined [50] as

$$\phi(l) = \sum_{k=1}^{l} N_{q(k)} \tag{54a}$$

$$\phi(0) = 0. \tag{54b}$$

If we attempt a DTW match between $T(m)$ and $R^S(n)$, and we define the resulting distance at the end of the match as $D_{q(1)q(2) \cdots q(L)}(M)$, then the "ideal" solution to the connected-word-recognition problem is the solution of the minimization

$$D^* = \min_{L} \left[\min_{q(1)q(2)\cdots q(L)} [D_{q(1)q(2)\cdots q(L)}(M)] \right] \tag{55}$$

i.e., the reference string, of length L words, which minimizes the accumulated distance is the best estimate of the test string.

It should be clear that, except for trivial cases, direct, exhaustive solution of (55) is impractical due to the excessive computation involved. For example for $L = 5$, $V = 10$ (i.e., a ten-word vocabulary with up to 5 words in a string), a total of

$$NS = 10^5 \text{ (five-word strings)} + 10^4 \text{ (four-word strings)} + \cdots + 10 \text{ (one-word strings)}$$

would have to be tested. As such there have been several approaches proposed for solving (55) in an efficient manner [8], [17], [18], [50], [51]. These approaches all attempt to reduce the computational loading by solving the minimization in a series of stages (levels) in which sufficient information is retained so that a series of string candidates is evaluated. Generally, the "best" string is retained as one of the candidates, and the vast majority of strings with poor distance scores are discarded.

It is impossible to discuss, in any detail, the different approaches to DTW algorithms for connected-word recognition. However, to illustrate the sophistication in these methods, we will discuss briefly the level building approach of Myers and Rabiner [50] as it serves to explain the general principles of operation of most of the methods.[4]

[4] The material in this section is taken directly from [19]. It has been shown that the level-building algorithm is related to the stack algorithm of Bahl and Jelinek [83].

1) The Level-Building Approach to Connected-Word Recognition: The manner in which the level-building algorithm is implemented for solving the minimization of (55) is illustrated in Figs. 25-26. Fig. 25 shows the simple case of obtaining the DTW distance between the test pattern $T(m)$, and a *given* super reference pattern $R^s = R_{q(1)} \oplus R_{q(2)} \oplus \cdots \oplus R_{q(L)}$, i.e., for fixed indices $q(1)q(2) \cdots q(L)$. A constrained endpoint DTW algorithm in which the slope of the warping function $w(m)$ is constrained to lie between ½ and 2 is used to find the best path within the parallelogram for matching T and R^s. This procedure could, in theory, be used to solve (55) by exhaustively testing every possible R^s and doing the minimization directly. However, as discussed previously, the amount of computation (V^L comparisons), even for modest values of L, is untractable.

In order to see how we can efficiently solve (55) we must examine, in more detail, the way in which the DTW algorithm is generally implemented for a fixed R^s and T. Fig. 26(a) shows a typical implementation of a constrained endpoint DTW algorithm. Generally, the computation to find the optimum warping path is performed in vertical stripes (i.e., m is indexed sequentially and a range on n is found in which the path is constrained to lie) as illustrated in this figure. An alternative way in which the computation could be performed is illustrated in Fig. 26(b). A set of horizontal lines has been drawn for different ending frames of the references within R^s. For this case the computation is done in vertical stripes again, however, the horizontal line formed by the end of each reference forms a constraint on the region in which the computing is done. As such, the computation is again done in vertical stripes until the partial region G_1 of the parallelogram is covered. In order to correctly pick up the computation for the second reference pattern (i.e., in region G_2) the accumulated distance scores for all paths that end at the first horizontal line (devoted by the heavy dots) must be retained, and used as initial conditions on distances. In this manner, the identical computation, as shown in Fig. 26(a) can be carried out by levels (i.e., words within the sequence of reference patterns) in a series of computations.

The significance of the above results is that the *level-building* approach to finding the best dynamic path (i.e., finding the best path for each reference pattern in the sequence) can be extended to the case of more than one reference pattern at each level, by computing all possible distances at the end of each level and retaining the best distance for each index m. Thus, if we define a range beginning variable $m_1(l)$ and a range ending variable $m_2(l)$, then in order to solve for the best match to the test pattern, for each value of m in the range $m_1(l) \leq m \leq m_2(l)$, at level l, we must keep track of three quantities, as follows.

1) Minimum accumulated distance, $\tilde{D}_l^B(m) = \min_{1 \leq v \leq V} [\tilde{D}_l^v(m)]$ where $\tilde{D}_l^v(m)$ is the accumulated distance for the vth reference pattern, at level l ending at frame m of the test pattern.

2) Best reference, $W_l(m) = \operatorname{argmin}_{1 \leq v \leq V} [\tilde{D}_l^v(m)]$, i.e., the reference pattern leading to the minimum accumulated distance.

3) Backtracking pointer, $\tilde{F}_l^B(m) = \tilde{F}_l^{W_l(m)}(m)$, where $\tilde{F}_l^v(m)$ is the frame of the test pattern at level $l-1$ at which

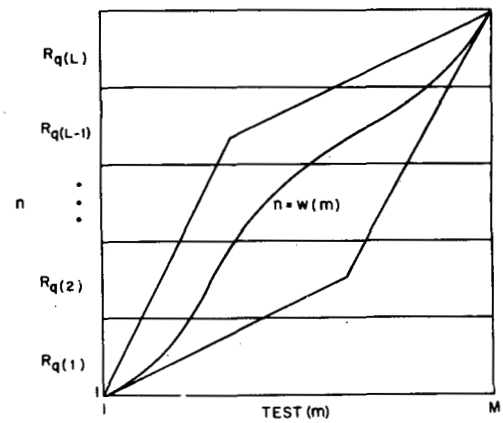

Fig. 25. Illustration of constrained endpoint DTW match of a sequence of reference patterns to a connected-word test string.

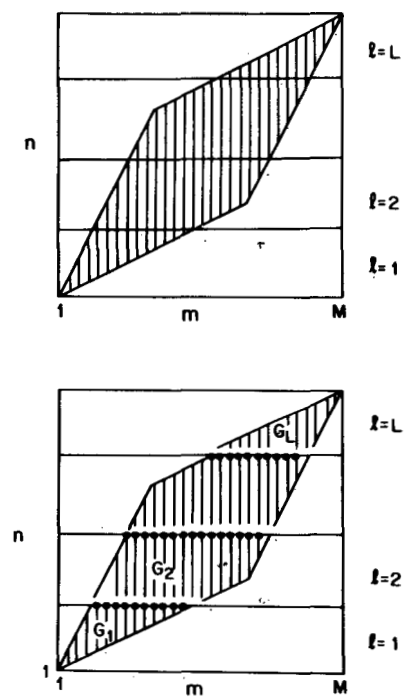

Fig. 26. Two possible implementations of constrained endpoint DTW algorithms.

the best path to test frame m, at level l, for reference pattern R_v, ended, i.e., the best path to frame m of the test pattern, at the end of the lth level using reference R_v, began at frame $\tilde{F}_l^v(m) + 1$. This pointer basically keeps track of the ending frame of each path at the previous level. For level $l = 1$, it should be clear that $\tilde{F}_l^B(m) = 0$ for all m, since all paths started at frame 1 of the test pattern.

Fig. 27 illustrates the operation of the level-building algorithm for a simple example where it is assumed that there are only two reference patterns, denoted as A and B, each of equal length. It is assumed that a string of length $L = 4$ is known to have been spoken. At the end of the first level, there are six possible ending values of m, and the reference pattern giving the smallest distance is denoted along the horizontal line at the end of the level. Similarly, at levels two and three best paths to each possible ending frame are noted by the refer-

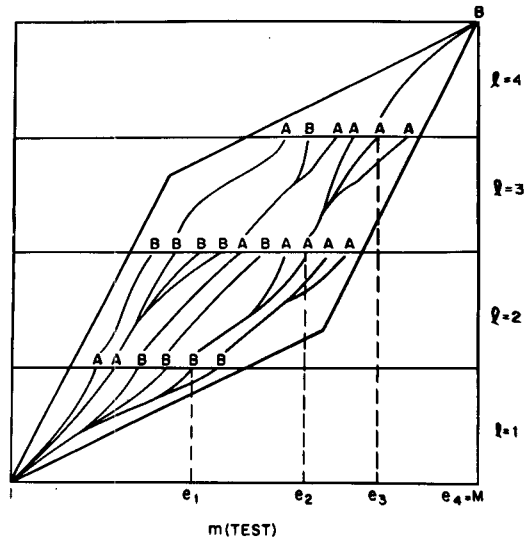

Fig. 27. Illustration of backtracking to recover the best sequence match in a four-level DTW match.

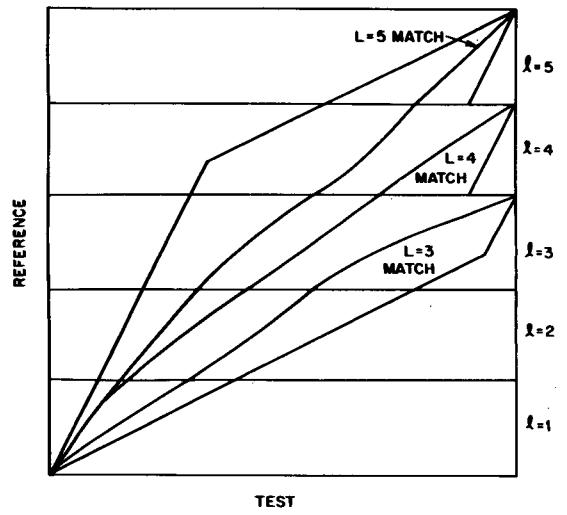

Fig. 28. Example of use of the level-building algorithm with several different length candidates.

ence, at that level, which gave minimum accumulated distance. Finally, at level four, only a single path is retained, as this is the optimum path which minimizes the distance of (55). To determine the best matching string, we must backtrack the path ending at $m = M$ to give the sequence BAAB as the best sequence of four reference patterns matching the test pattern. Also denoted in Fig. 27 are the test frame values e_l corresponding to the end of each reference in the best matching sequence. In principle, these values e_l could be used as best estimates of segmentation points between entries in the test pattern.

It should be noted that the level-building algorithm, as presented above, is capable of determining the best matching string to a test pattern of variable length. As such, the algorithm can generate several "best" matches, each of different lengths as shown in Fig. 28. The overall "best" match is defined as the match giving the smallest distance over all possible sequence lengths. The alternative length strings are useful for applications in which the length of the string is known *a priori*, e.g., telephone number dialing, credit-card codes, etc. In Fig. 28, we show best matches for strings of length $L = 3$, 4, and 5, for the given example.

A second point of note is that, by doubling the storage at each level, we can keep track of both a "best" path, and a second best path, to each frame m of the test pattern. In this manner, alternative estimates of reference strings can be estimated by using second best paths at any level in the warp. This important point is illustrated in Fig. 29, which shows a "best" path for an $L = 4$ length string, and a series of four alternative paths obtaining by substituting a second best distance alternative at each level in the warp. These paths are shown graphically in Fig. 29(a), and symbolically in Fig. 29(b). If we denote the best path by the sequence of arcs 1111, then the alternative paths 2111, 1211, 1121, and 1112. However, the arcs labeled 1's occurring before an arc labeled 2 in the alternative paths, need not be the same arcs labeled 1 for the best path, since we are now finding a best path to a different ending frame at each level. The set of distances associated with each of these suboptimum paths can be ordered to give an alternative list of strings as estimates for the spoken string.

Fig. 30 illustrates the application of the level-building algorithm to connected digit strings. In this plot, we show the log energy contour of the spoken string [part (a)] and a series of plots of the accumulated distance for each digit at each level. Fig. 30 is for the spoken string 51560. At the end of each level, the program prints out the best local estimate of the digit at each level (shown to the right of each level rectangle). The verticle dashed lines, at each level, denote the initial range of m for which the level allows paths to begin. The sloped dashed line at each level is a distance rejection threshold (similar to the one used for isolated words) to eliminate candidates which accumulate large distances. For this example, the best estimate, at each level, of the spoken digit is the actual spoken digit. At the end of the fourth level, the string 5157 matched to the end of the test string with an average distance score of 0.533. Three alternative strings, namely, 1157, 5957, and 5117, were also generated at this level by using the second-best distance candidates at each position in the string. At the fifth level, the string 51560 (the correct one) was obtained with an average distance score of 0.333. Alternate choices, at this level, included the strings 11560, 51570, 59560, 51160, and 51562 with average distance scores as shown on the figure.

The level-building DTW algorithm can be easily modified both to increase its efficiency and to increase its flexibility in handling various types of test input strings. To accomplish these tasks, a set of variables has been defined that can be independently controlled, and which influence the performance of the level-building algorithm. These variables include the following.

1) δ_{R_1} = region of uncertainty at the beginning of the reference pattern.

2) δ_{R_2} = region of uncertainty at the end of the reference pattern.

3) δ_{END} = region of uncertainty at the end of the test pattern.

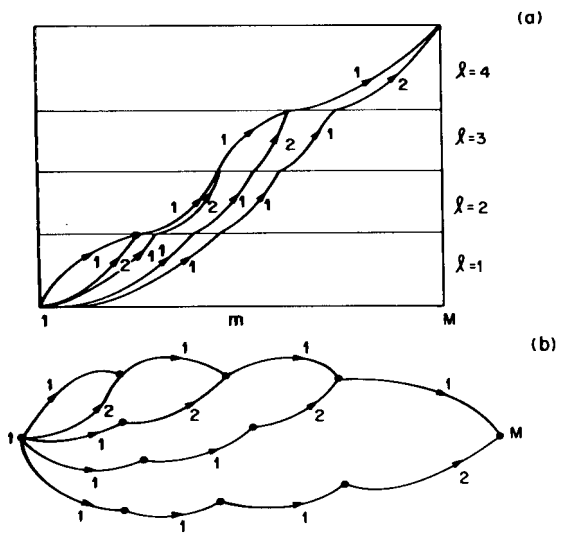

Fig. 29. Example illustrating how additional candidate strings are obtained in the level-building algorithm.

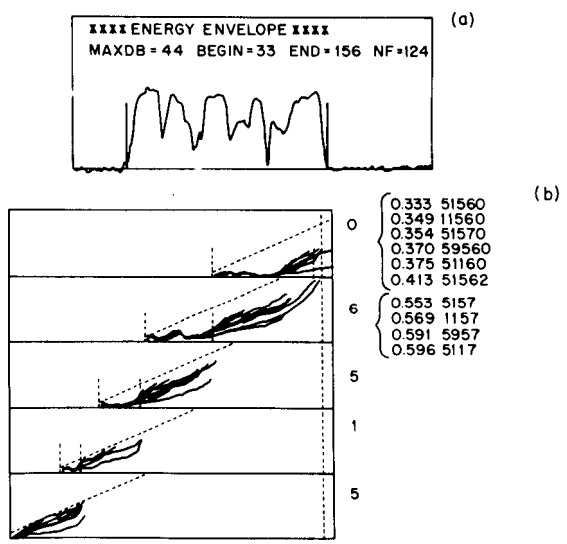

Fig. 30. Log energy contour and accumulated distance at each level of the level-building algorithm for the test string 51560.

4) M_T = distance multiplier to determine initial m-axis range, at each level, along the test pattern.

5) ϵ = range of DTW local warp along the reference pattern.

6) T_{min}, T_{max} = threshold parameters on accumulated distance at each level.

7) KNN = K-nearest neighbor decision rule criterion.

The physical interpretations of some of these variables on the level-building paths are illustrated in Fig. 31. The variables δ_{R_1} and δ_{R_2} define regions, at the beginning and end of each reference pattern, in which the local path can begin or end, i.e., paths need not begin at frame 1 of each reference and end at the last frame, but instead the best beginning and ending frames, within the specified regions, are found and used for each path.

Similarly, the parameter δ_{END} defines a region at the end of the test pattern in which a total match can end, rather than strictly requiring each path to end at that frame $m = M$. This added flexibility allows for some margin of error in determining the ending frame of the test pattern.

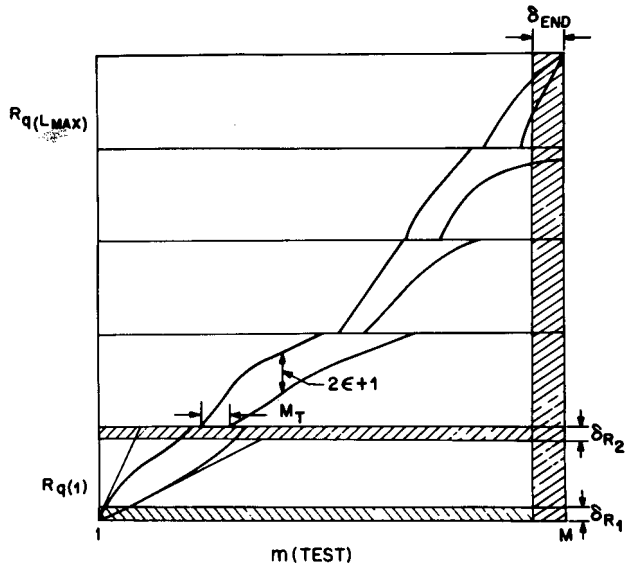

Fig. 31. Illustration of the level-building parameters δ_{R_1}, δ_{R_2}, δ_{end}, M_T, and ϵ.

The parameters M_T and ϵ are range reduction parameters which reduce the size of the local regions G_l at level l, in which the dynamic path is constrained to lie. The parameter M_T is used to reduce the initial starting range (i.e., from $m_1(l)$ to $m_2(l)$) to the reduced range

$$S_l^1 \leqslant m \leqslant S_l^2 \tag{56}$$

where S_l^1 and S_l^2 are defined as

$$S_1^1 = \text{largest } m \text{ such that } \tilde{D}_{l-1}{}^B(m)/m > M_T \cdot \phi_{l-1} \quad \text{for all } m < S_l^1 \tag{57a}$$

$$S_l^2 = \text{smallest } m \text{ such that } \tilde{D}_{l-1}{}^B(m)/m > M_T \cdot \phi_{l-1} \quad \text{for all } m > S_l^2 \tag{57b}$$

where

$$\phi_{l-1} = \min_{1 \leqslant m \leqslant M} \left[\frac{\tilde{D}_{l-1}{}^B(m)}{m} \right] \tag{58}$$

is the minimum average distance at the end of level $l-1$. For practical values of M_T, the range of starting values of m can be reduced by about 50 percent.

Similarly, the parameter ϵ defines a range of width $2\epsilon + 1$ for searching in the n-direction, to find the best path for each reference at each level, as defined by Rabiner *et al.* in the UELM DTW algorithm [35]. At each frame m, along the test, the range along n is determined by examining a region within $\pm\epsilon$ frames (along n) of the minimum accumulated distance at frame $m-1$. This parameter is again primarily used to reduce the size of the search region.

The parameters T_{min} and T_{max} are used to terminate DTW searches on reference patterns which accumulate excessive incremental distance at any level l. Details of the incremental distance test are given in [16].

The last method for increasing flexibility of the level-building algorithm, namely, the use of the K-nearest neighbor rule (KNN) for speaker-independent recognition, is a difficult one to implement. This is because the KNN rule assumes that the distance scores which are being compared (and averaged) all are generated using the same test pattern. For isolated-word recognition, this assumption is valid; however, in the level-building algorithm, the best paths at level l and frame m, from two templates representing the same word, can begin at different starting frames. Hence, these two paths use different portions of the test string and cannot be averaged in a KNN rule. Thus, in general, it is not possible to compare distance scores directly with the KNN rule. However, a reasonable heuristic for the KNN rule can be applied, namely, that the distance scores from two reference templates (representing the same vocabulary word) may be averaged in a KNN rule if both templates have warping paths that "came from" the same word. To implement the revised KNN rule, for each token of the vth vocabulary word, we must keep track of the distance accumulated to frame m, and a pointer of the word (at the previous level) from which the best path began. For all frames where KNN (or more) tokens are defined to have come from the same word at the previous level, the KNN rule averages the KNN smallest distances, to give the word distance at frame m and level l.

2) Results on the Recognition of Connected-Digit Strings [19]: The level building DTW algorithm of the previous section was applied to the task of connected-digit recognition and the results of a series of performance evaluations are given in Table V. For these tests, a total of six talkers (three male, three female) each spoken 80 connected-digit strings. The strings varied in length from two to five digits, and were randomly chosen, but balanced to have an equal number of occurrences of each digit in each string length, on average. A total of 20 strings of each length were spoken. The strings were recorded over dialed-up telephone lines.

The recognizer was tested as both a speaker-trained system (one template per word) and a speaker-independent system (using the clustered, isolated-word templates described previously). Table V(a) shows the error rate scores for string errors (i.e., if any digit in the string was wrong, the whole string was in error), word errors, and string errors with "known length", i.e., only strings of the correct length were allowed. It is seen that string error rates of about 4.7 percent and word error rates of about 0.8 percent are attained for *both* speaker-trained and speaker-independent systems.

Table V(b) shows a breakdown of the string errors by talker (M for male, F for female). It is seen that the speaker-trained system performed better for males than for females, however, the speaker-independent system performed equally well for both males and females.

3) Summary: The area of connected-word recognition is one which is being investigated actively at a number of research laboratories. It is anticipated that as more experience and insight is gained into the strengths and weaknesses of the connected-word DTW algorithms, further applications of this promising technology will appear.

TABLE V
RESULTS ON CONNECTED-DIGIT RECOGNITION USING THE LEVEL-BUILDING ALGORITHM. (a) SIX TALKERS–THREE MALE, THREE FEMALE; 80 STRINGS PER TALKER; 20 EACH OF LENGTH 2, 3, 4, 5, DIGITS; BALANCED DIGITS WITHIN STRINGS. (b) NUMBER OF STRING ERRORS.

	STRING ERROR RATE	WORD ERROR RATE	STRING ERROR RATE-KNOWN LENGTH
SPEAKER TRAINED (Single Template Per Word)	4.8%	0.7%	3.8%
SPEAKER INDEPENDENT (12 Templates Per Word)	4.6%	0.9%	3.5%

(a)

TALKER	SPEAKER TRAINED	SPEAKER INDEPENDENT
LR (M)	3	1
JG (M)	3	7
SL (M)	2	3
KS (F)	6	3
CS (F)	3	5
SC (F)	6	3

(b)

IV. THE ROLE OF GRAMMAR AND SEMANTICS IN SPEECH RECOGNITION

In the foregoing discussion, automatic speech recognition has been treated as a classical pattern-recognition problem. The techniques presented all rest on the almost universally accepted fact that the intelligence in speech is largely encoded in the temporal variation of the short-time amplitude spectrum. It was, therefore, appropriate to have discussed methods for estimating spectra, tracking their variation in time and comparing them to prototypical patterns (reference templates). Armed with these methods and supported by the ever increasing computational power provided by a variety of electronic technologies, one can build robust speech-recognition devices which perform practical human/machine communication tasks on an economically competitive basis.

Not surprisingly, however, such machines are limited in their ability to effect versatile, fluent speech communication simply by extrapolation of the above discussed techniques. The reason is , of course, that speech is a complex code in which the intelligence of a message is represented both locally, by spectral features, and globally by a hierarchy of structural features. If we are to build devices which function with the richness and robustness of natural speech communication, the structural aspects must be considered. It is important to note, in passing, that this fact was appreciated by the earliest researchers of speech recognition (see, for example, Denes [52]). It is only now, however, that we have both the mathematics and the computers to model and study some of the global aspects of the speech code. Many of the modern speech-recognition systems rely heavily on the structural and linguistic aspects of speech [53]-[56], [28], [20], [21].

In this section, then, we shall investigate two levels of the hierarchy of structural information, grammar and semantics, and their application to mechanical speech recognition. For the purposes of our discussion, grammar is the surface structure of a message and includes, but is not limited to the phonetic structure of words and word order in sentences. Semantics is the deep structure of a message by which meaning is conveyed.

Let us begin by placing grammar and semantics in perspective with respect to the speech-communication process. Speech is a code used to convey information. Peirce [57] has distinguished among four aspects of natural language codes, symbolic, syntactic, semantic, and pragmatic. The symbols of a language are arbitrary and differ both from language to language and from the written to the spoken form of a given language. For written English, for example, the symbols are the 26 letters of the alphabet, a blank symbol or a space, and a few punctuation marks. For spoken English, the 40 or so basic sounds or phonemes are a reasonable choice. Although they are subject to substantial variation, they do correlate highly with measurable spectral parameters.

Syntax is the relationship of symbols to each other. Although we usually think of syntax as grammar, that is the way the words are concatenated to form sentences, syntax equally well describes the way spectral types form phonemes, phonemes form syllables, and syllables form words. The syntatic structure of a language is also arbitrary to the extent that any set of rules for forming sequences of symbols is legitimate so long as the sequences can actually be realized. In speech, for example, one would not expect to find sequences of phonemes which are anatomically impossible to articulate. Later we shall see that while syntatic rules are arbitrary, some sets are better than others in the sense that they provide for more robust communication.

Semantics is the relationship of symbols to reality. It is at this level of the communication hierarchy that the arbitrariness ends. Once certain symbols are chosen to represent specific aspects of the real world, certain constraints on the way symbols are arranged in sequences are automatically imposed if we are to have a faithful linguistic model of our universe.

The boundary between syntax and semantics is easily emphasized by an example. Consider the sentence: *The green colorless cloud pulled coldly on the fast shirt.* According to the rules of English grammar, the sentence is well formed. It is, however, clearly semantically anomalous and in ordinary conversation, would not be understood. The semantic constraints inherent in natural language, like those imposed by syntax, serve to make it more robust and must be exploited for the same purpose in constructing speech-recognition machines.

Pragmatics is the relationship between symbols and their users. Two different speakers, or the same speaker in two different contexts, will use the same symbol to mean entirely different things. This aspect of language is very difficult to formalize and will not be treated in this paper. The definition is given only for the sake of completeness.

Having defined the fundamental aspects of communication by natural language, we turn to the human speech-communication process to see how these fundamentals are manifest. We shall later use this understanding as a guide to the design of a speech-recognition machine.

Referring to Fig. 32, let us follow the processes which occur in the course of speech communication. A speaker wishing to convey a message must first formulate it in accordance with the semantic structure of the language. The conceptual representation must then be transformed into the corresponding linguistic encoding. Finally, neuromuscular activities must be coordinated so that the lungs, vocal cords, and articulatory organs of the vocal tract move in the proper sequence to produce the sound pressure wave which can be heard as speech.

The hearing process begins in the peripheral auditory system, where spectral analysis is performed by the exquisitely sensitive and selective filter bank called the cochlea. Spectral and temporal information is quantized, encoded, and transmitted via the auditory nerve to the auditory cortex of the brain where symbols are detected, the linguistic code is deciphered and the semantic content extracted whereupon the message is understood.

Even from this oversimplified account of the speech-communication process, several insights which have profound effect on our approach to human/machine speech communication may be derived. The first is the change in data rates which occurs at each stage in the process. The bit rates required after each transformation are shown in Fig. 32. These figures are derived from the "text to speech" synthesis system of Coker [58] and are, of course, only estimates. They serve, however, to empahsize the important fact that the speech signal is highly redundant. Clearly a mechanical speech recognizer should exploit this redundancy.

It is also apparent that speech communication involves cognitive processes which implies that the human capacity for speech communication is inextricably related to our intelligence. We should, therefore, not expect to achieve high-quality speech-recognition machines until we can simulate human intelligence.

One distinctive feature of intelligence is the transition from processing continua to manipulating discrete symbols. While the boundary is not always clearly demarked, its presence is unmistakable. In the speech-production process the change occurs somewhere between linguistic encoding and muscular actions. For the listener, the transition lies somewhere between the cochlear mechanics and linguistic decoding.

It is entirely appropriate that this paper reflect this characteristic transition. The first three sections have dealt with the signal processing and pattern-recognition aspects of speech recognition and were made precise in terms of classical applied mathematics. The formalisms used in the remainder of the presentation will be predominantly those of discrete, strictly nonnumerical computation. We shall encounter one striking exception which serves as a bridge between the two worlds and reminds us of their vague common boundary.

As for the speech-production mechanism, we can make rather faithful mathematical models of the process. In these representations there is a clear and intuitively appealing relationship between reality and abstraction. Coker *et al.* [59]

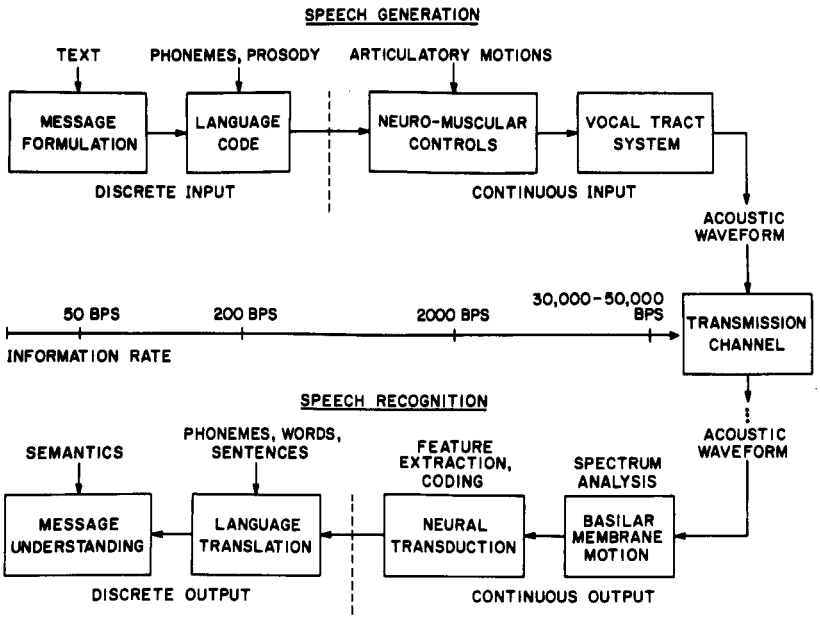

Fig. 32. Schematic representation of the human speech-communication process.

have built a speech-synthesis system in which messages in the form of ordinary text are linguistically encoded using a pronunciation dictionary and a table of prosodic rules for pitch, intensity and duration. The linguistic code is then transformed into a sequence of articulatory motions and the response of the vocal tract to an appropriate excitation fucntion is computed resulting in intelligible synthetic speech. Flanagan *et al.* [60] have investigated the dynamics of the vocal cords and have coupled it with an even more detailed vocal tract model in an effort to gain even more understanding of the physical mechanisms involved in speech production.

Unfortunately, the same kind of model of the mechanics of speech recognition is not available to us. We resort instead to a metaphorical model in which abstract formalisms are used instead of physical and physiological correlates. In fact, the formalisms which we use are the very simplest forms of a very rich mathematical discipline. Despite their obvious shortcomings, our models further our efforts to build speech-recognition machines in two ways. First, they exhibit some of the important phenomena observed in natural speech communications. In particular, they yield a explanation for and quantification of the simultaneous increases in versatility and robustness which are obtained by the addition of grammatical and semantic analyses. Second, they enable us to actually build more powerful speech-recognition devices. Thus we have pursued these models because of their analytical and experimental value.

A. Syntatic Analysis

We begin our discussion of higher level processing in speech recognition by considering systems of the form shown in Fig. 33. These systems operate as a two-level hierarchy in which the first level is an acoustic processor and the second, a language analyzer. Throughout this discussion, the acoustic processor will be assumed to be the isolated word recognizer described in Section II above. The language analyzer can be any of a number of devices, several of which will be described in this section.

1) Rudimentary Linguistic Analysis: Regardless of what the specifics of the language analyzer are, however, its conceptual description is the same. Suppose that the acoustic processor is capable of recognizing a vocabulary of a finite numer of words with a small but nonnegligible probability of error. Imagine that we wish to recognize messages composed of sequences of the vocabulary words (with clearly marked boundaries between words[5]) and that we have a list of all the messages of interest. When a message is processed acoustically, the transcirption may be corrupted (due to errors in either production or recognition). The language analyzer takes this possibly incorrect string of symbols and finds the message in the list which most closely matches, in a well-defined sense, the acoustic transcription. This message is then taken to be the actual utterance.

If the user of such a system is cooperative in the sense that he only utters sequences which are actually in the list, and if we have made the number of messages in the list smaller than the number of arbitrary sequences of vocabulary words, then we would expect the process to be able to correct errors in the acoustic transcription. At the same time, we may increase the versatility of the speech-recognition system by requiring that there be more messages than vocabulary words.

In order to implement the procedure exactly as described above, one need only define the best-match criterion used in searching the message list. Toward that end, we assume an n word vocabulary V_T where

$$V_T = \{v_j\}_{j=1}^{n}. \tag{59}$$

Since the acoustic processor recognizes a word by computing the distance from the unknown utterance to the prototype for

[5] Often the input is a sequence of isolated words with distinct pauses between words.

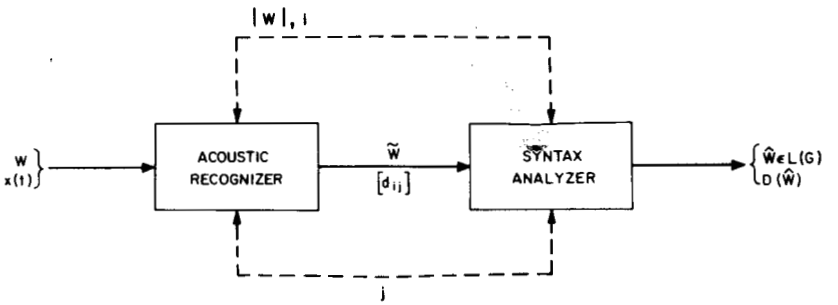

Fig. 33. Block diagram of two-level speech-recognition system.

each word in the vocabulary and then applying some decision rule, for an input string W consisting of the k words $v_{j_1} v_{j_2} \cdots v_{j_k}$, i.e.

$$W = v_{j_1} v_{j_2} \cdots v_{j_k} \tag{60}$$

the acoustic processor must compute all the distances for each of the k words. It is convenient to store these distances as a matrix,

$$D = [d_{ij}] \, 1 \leqslant i \leqslant k; \qquad 1 \leqslant j \leqslant n \tag{61}$$

in which d_{ij} is the distance from the ith word in the string v_{j_i} to the prototype for the jth vocabulary word v_j.

To describe the simplest form of the search procedure we denote the message list by L where

$$L = \{W_i\}_{i=1}^{N} \tag{62}$$

and each W_i is a string of the form of (60). If

$$W_i = v_{j_{i_1}} v_{j_{i_2}} \cdots v_{j_{i_k}} \tag{63}$$

then its distance $D(W_i)$ is given by

$$D(W_i) = \sum_{l=1}^{k} d_{l j_{i_l}} \tag{64}$$

which we can compute for $1 \leqslant i \leqslant N$. Then ordering the list of distances so that

$$D(W_{m_1}) \leqslant D(W_{m_2}) \leqslant \cdots \leqslant D(W_{m_N}) \tag{65}$$

the input W is recognized as W_{m_1}. The obvious drawback to this method is that the amount of computation grows linearly with N. For many otherwise practical tasks, this is prohibitive.

The computational burden could be greatly reduced if we could enumerate strings of symbols in order of their distances. When a string X is found such that $X \in L$, then the search terminates and $W = X$. An efficient procedure for so enumerating the strings has been devised by Aho *et al.* [61].

The algorithm of Aho *et al.* for enumerating the Cartesian product of ordered sets is as follows. Sort the matrix D on its second subscript so that for each i, $1 \leqslant i \leqslant k$ we have

$$d_{ij_1} \leqslant d_{ij_2} \leqslant \cdots \leqslant d_{ij_n}. \tag{66}$$

Note that in this process we have ordered the vocabulary symbols with $v_{j_{i_1}}$ having the highest likelihood. Clearly then the string of minimum distance $\hat{W}_1$ is just $v_{1j_1} v_{2j_1} \cdots v_{kj_1}$ and its distance D_1 is given by

$$D_1 = \sum_{i=1}^{k} d_{i1}. \tag{67}$$

The key observation is that if $\hat{W}_k$ is the kth most likely string, having distance D_k, then the $k+1$st most likely string, $\hat{W}_{k+1}$, differs from one of $\hat{W}_1, \hat{W}_2 \cdots \hat{W}_k$ in only one position and in that one position the second subscript advances by exactly 1. So, for example,

$$W_2 = v_{1j_1} v_{2j_2} \cdots v_{l-1,j_1} \cdots v_{lj_2} v_{l+1,j_1} \cdots v_{kj_1} \tag{68}$$

for some l such that $1 \leqslant l \leqslant k$. The corresponding distance D_2 is just

$$D_2 = \left[\sum_{\substack{i=1 \\ i \neq l}}^{k} d_{i1} \right] + d_{l2}. \tag{69}$$

Comparing (69) to (67) we see that the right-hand sides differ by only one term, the lth, and in that term the second subscript differs only by one.

The above discussion suggests the following simple algorithm. Given D_m for some m, compute all string distances which differ from D_m in only one term and, in that one term term, have a second subscript which is larger by one. Store these distances in a sorted list or priority queue. Delete D_m from the queue. The smallest entry remaining on the list is D_{m+1} and the corresponding string is easily found if the subscripts are maintained on a parallel queue.

The priority queue can be implemented in the form of a binary tree as described by Knuth [62]. Using this data structure, only $M \log_2 M$ operations are required to find the M smallest string distances. The tree need have at most M nodes so that the storage is not excessive. Of course, we have no assurance that one of $\hat{W}_1, \hat{W}_2, \cdots, \hat{W}_M$ is actually in L unless $M = n^k$. So if we restrict M to some tractably small size, we must be prepared for search failures. There are, of course, other methods for searching L. One is based on organizing L as a tree and applying sequential decoding techniques [85].

Aldefeld *et al.* [49] have derived a method which combines the certainty of the exhaustive search technique with the efficiency of the Cartesian product algorithm. The underlying idea is to impose an equivalence relation, $\equiv$, on V_T. One way to do this is to let $v_i \equiv v_j$ whenever v_i and v_j are acoustically similar. The algorithm will still work however if membership in equivalence classes is assigned arbitrarily. Let us denote the number of classes by N_V.

Once defined, we can extend the equivalence relation to strings of vocabulary words by letting

$$v_1 v_2 \cdots v_k \equiv v_1' v_2' \cdots v_k' \quad \text{iff } v_j \equiv v_j' \text{ for } 1 \leqslant j \leqslant k. \tag{70}$$

Based on the classification of letters and strings we may define string-class distances. Let $h(v_1 v_2 \cdots v_k)$ be the set of all strings equivalent to $v_1 v_2 \cdots v_k$. Then the distance of $h(v_1 v_2 \cdots v_k)$, $D(h(v_1 v_2 \cdots v_k))$, is defined by

$$D(h(v_1 v_2 \cdots v_k)) = \min \{D(v_1' v_2' \cdots v_k') \mid v_1' v_2' \cdots v_k' \equiv v_1 v_2 \cdots v_k\} \tag{71}$$

meaning that the string-class distance is the least distance of any of its members. Equation (71) can be easily evaluated according to

$$D(h(v_1 v_2 \cdots v_k)) = \sum_{i=1}^{k} \min \{d_{ij_i} \mid v_{j_i} \equiv v_i\}. \tag{72}$$

The new search algorithm is now easily described. Given a D matrix, the string class distances are computed in increasing order by the Cartesian product algorithm. Each class is searched by the exhaustive search technique for the string within L which has the minimum distance. When the next class distance exceeds the distance of the best string found thus far, the algorithm terminates and the current best string is the string of minimum distance. Maximum efficiency of the algorithm is achieved by selecting a value of N_V so that a balance is maintained between the number of string classes and the number of strings in each class.

The list searching method of language analysis has been implemented and has proven to be very effective for the problem of retrieving telephone directory information by spoken spelled queries.[6] In the first of several experiments, Rosenberg and Schmidt [45] used a vocabulary consisting of just the 26 letters of the alphabet and a message list in the form of the 18 000 entry Bell Laboratories telephone directory. Using a heuristic search procedure they obtained 92 percent correct directory information with an acoustic process of 79 percent accuracy on individual letters using templates obtained from each individual talker.

Subsequently, the equivalence class algorithm was implemented and tested on the same problem by Aldefeld *et al.* [49]. A block diagram of the system has already been given in Fig. 21. For convenience the vocabulary was partitioned into two classes C_1 and C_2 where

$$C_1 = \{B, C, D, E, G, O, P, Q, T, U, VW, Z\} \tag{73a}$$

$$C_2 = \{A, F, H, I, J, K, L, M, N, R, S, X, Y, \text{blank}\} \tag{73b}$$

resulting in 256 string classes, for strings of eight words, with an average of 70 strings per class. The telephone directory was then reorganized, storing equivalence classes sequentially on a disk file.

Again using an acoustic processor with an error rate of 20 percent on letters, a 1.4 percent error rate on directory retrievals was measured. On the average, only 1.2 percent of the directory was searched for a query. As discussed previously, these results show that some very simple linguistic processing can greatly enhance our speech-recognition capabilities.

B. Applying Formal Language Theory to Linguistic Processing

The list search technique described above has an obvious limitation. It will not work if there is no explicit list of messages. Unfortunately, this is exactly the case in Natural Language communication. There is no explicit list of all the possible sentences of English, for example, which might occur. What people seem to have, however, is a set of rules which enables them to generte whatever part of the list is needed in a given conversation. In this section, we shall see how the essence of this process is captured by elementary formal language theory and how this theory can be advantageously applied to speech recognition.

1) Definitions and Nomenclature: We shall use the symbol $L(G)$ to denote a language generated by a grammar G. The grammar is a function of four arguments: $G = G(V_N, V_T, S, P)$. As in the case of lists of messages discussed in the previous section, V_T is a finite set of symbols out of which sentences are composed. V_N is another finite set of symbols disjoint with V_T and whose members never appear in sentences. $S \in V_N$ is a distinguished symbol called the start symbol. The heart of the grammar is a finite set of transformations called production rules and represented by P. The transformations map strings of symbols from V_N and V_T into other such strings. This is written as

$$\alpha \rightarrow \beta; \alpha, \beta \epsilon \{V_N \cup V_T\}^* \tag{74}$$

where the asterisk means the set of all strings of elements of the designated set.

A string of symbols $W = w_1 w_2 \cdots w_k$ where $w_i \in V_T$ for $1 \leqslant i \leqslant k$ is said to be a sentence in the language, written $W \in L(G)$, if and only if there exists a sequence of production rules wich map S into W. That is

$$S = > \alpha_1; \alpha_1 = > \alpha_2; \cdots; \alpha_r = > W \tag{75}$$

which we signify by $S \stackrel{*}{=} > W$.

Different forms of production rules lead to languages of different complexity. Chomsky [63] has formulated a taxonomy of languages, often referred to as the Chomsky hierarchy, based on the form of the production rules required for genera-

[6] The reader may want to refer to Section III-B for a general discussion of the directory listing retrieval system.

tion. From this seminal work has developed a very beautiful and important branch of mathematics called formal language theory which is essential to the study of both natural languages and artificial ones such as programming languages. In this section, we shall look briefly at one tiny aspect of formal language theory which has a bearing on speech recognition. Readers interested in further study of formal language theory should consult Hopcroft and Ullman [64].

If we restrict the form of production rules to

$$A \rightarrow aB \tag{76a}$$

or

$$C \rightarrow b \tag{76b}$$

where $A, B, C \in V_N$; $a, b \in V_T$, we get the set of Chomsky type 3, or regular, grammars. Linguists are fond of reminding us that this simplest of formal grammars does not have the richness of structure to generate a natural language. It does, however, allow us to generate nontrivial subsets of natural language appropriate to a specific topic.

Regular languages have the additional advantage of a graphic representation of the type shown in Fig. 34. This particular graph is called a state transition diagram and can be used in both a generative and analytical mode. Each transition from one state to another corresponds to a production rule in the grammar G.

The state diagram can be used to generate sentences in the language by beginning in state 1 and making transitions from state to state until a final state, denoted by an asterisk, is reached. As each transition is made, the vocabulary word associated with it is appended to the string formulated thus far. It is not customary to use the state diagram in this generative sense but rather to use P as the generator of the language.

The traditional use for the state transition diagram is to parse a sentence W, that is to determine if $W \in L(G)$, and if so what sequence of production rules form the derivation $S \overset{*}{\Rightarrow} W$. In terms of the state diagram, parsing is searching the graph for a path, beginning at state 1 and terminating at some final state, the labels on whose transitions, read in sequence, are the string under analysis. If no such path exists, then the string, W does not parse and we write $W \notin L(G)$. As we shall see presently, there are efficient ways of performing the search. Later on we shall discuss the significance of the path or state sequence corresponding to a given sentence.

For the moment, let us revisit Fig. 33 in the context of the present discussion. The acoustic processor operates exactly as before producing the distance matrix D. The language analyzer, in this case, finds the sentence $\hat{W} \in L(G)$ whose total distance

$$D(\hat{W}) = \sum_{i=1}^{k} d_{ij_i} \tag{77}$$

is minimized. Then $\hat{W} = v_{j_1} v_{j_2} \cdots v_{j_k}$ and the procedure for finding it and its derivation is called a maximum-likelihood parsing technique. This technique, too, can be described in

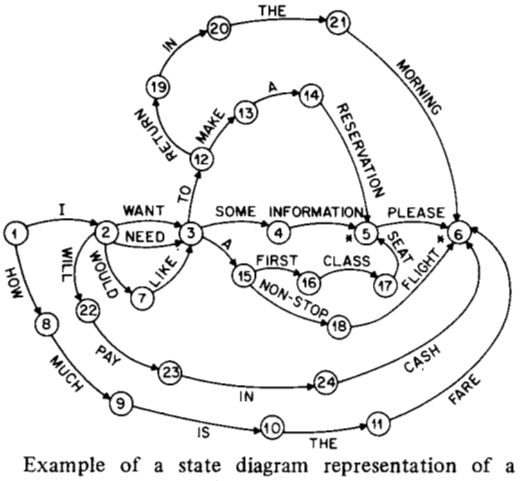

Fig. 34. Example of a state diagram representation of a regular language.

terms of the graph searching explanation of parsing given above. One simply assigns the appropriate entry of D to each state transition and then searches for the shortest path, i.e., the one of minimal total distance, from state 1 to a final state having exactly one transition for each row of D.

This search can be performed efficiently by a dynamic programming technique due to Dijkstra [65]. Let $Q = \{q_1, q_2, \cdots, q_N\}$ be the set of states in the state transition diagram. We then use two matrices $[\Phi_{iq}]$ and $[\Psi_{iq}]$ with $0 \leqslant i \leqslant k$, where there are k rows in D, and $1 \leqslant q \leqslant N$.

Initially we set

$$\Phi_{iq} = \begin{cases} 0 & \text{for } q = 1; i = 0 \\ \infty & \text{otherwise} \end{cases} \tag{78a}$$

and

$$\Psi_{iq} = 0 \qquad \forall i, q. \tag{78b}$$

The entries of $[\Phi_{iq}]$ and $[\Psi_{iq}]$ are computed recursively according to

$$\Phi_{iq} = \min_{\Delta} \{\Phi_{i-1,s} + d_{ij}\} \tag{79}$$

where Δ is the set of all transitions from state s to state q labeled by v_j. Then

$$\Psi_{iq} = \Psi_{i-1,s^*} v_{j^*} \tag{80}$$

where s^* and j^* minimize the right-hand side of (79).

If we define z to be the final state for which Φ_{kz} is minimum, then

$$\hat{W} = \Psi_{kz} \tag{81a}$$

and

$$D(\hat{W}) = \Phi_{kz}. \tag{81b}$$

The state sequence corresponding to $\hat{W}$, say $\bar{q}$, can be found using a method described by Levinson [66]. The significance of $\bar{q}$ will become clear from our discussion of semantic information processing.

The algorithm described above has been implemented and tested in a speech-recognition system by Levinson *et al.* [22]. The system uses a 127-word vocabulary designed for an airline timetable information and reservation task. Syntatic analysis is based on a regular grammar whose state diagram has 144 states and 450 transitions. The language generated by the grammar contains over 6×10^9 sentences. Using an acoustic processor having an 11.7 percent error rate on words, an overall word error rate after parsing of 0.4 percent was obtained. This corresponds to a sentence error rate of 3.9 percent for sentences with an average length of seven words. This result should be compared with a sentence error rate, without the parsing stage, of 58 percent!

By means of a computer simulation, Levinson [66] has investigated the error correcting properties of maximum-likelihood parsing algorithms. The result of a simulation based on the airline task language is shown in Fig. 35. Shown is a plot of word error probability as a function of the simulation control parameter σ. This parameter controls the variance of the probability density function for the distance from a test utterance to a prototype for that word. As σ increases, therefore, the vocabulary words are made to appear less distinct acoustically.

The upper curve shows that as σ increases, the error rate for acoustic recognition grows rapidly until it reaches its asymptotic value of $(1 - (1/127))$ or 0.992, the rate which would be obtained by random selection. The lower curve shows the error rates obtained when acoustic recognition is supplemented by maximum-likelihood parsing. The degradation in recognition accuracy is greatly retarded, especially in the region of realistic acoustic error rates from 5-10 percent.

Similar simulations on several different languages, including some more complex than the simple type 3 variety, have been performed by Levinson *et al.* [67]. The same qualitative behavior as is shown in Fig. 35 was always observed. There were, however, differences in asymptotic error rates and slopes of the curves. These differences are a function of some intrinsic properties of the languages under consideration.

In order to gain a more quantitative understanding of the differences between languages and how these differences affect the error correcting capability of maximum-likelihood parsing, we shall resort to the statistical properties of formal languages. In an early paper, Shannon [68] observed that readers of English could accurately guess at letters missing from text and he related this predictability property of language to a precise statistical measure of information, entropy. Shannon devised a number of methods for estimating the entropy of natural language. Following Shannon [84], Sondhi and Levinson [69] have shown how the entropy $H(L(G))$ of the regular language $L(G)$ may be exactly determined from its state-transition diagram.

The key quantity characterizing the state diagram is its connectivity matrix, defined by

$$c_{ij} = \text{number of transitions from } q_i \text{ to } q_j \tag{82}$$

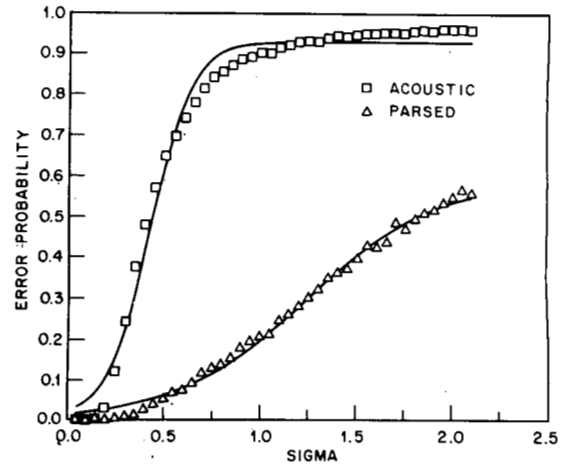

Fig. 35. Stimulation results showing the effects of maximum-likelihood parsing.

and $1 \leqslant i, j \leqslant N$. We can use the connectivity matrix to compute the number of sentences, N_k, of length k in $L(G)$ from

$$N_k = \vec{e}_1 C^k \vec{f}^T \tag{83}$$

where $\vec{e}_1$ is the first N component unit vector and $\vec{f}$ is an N vector of 1's and 0's whose ith component is 1 iff q_i is a final state. The superscript T signifies matrix transposition. Then, since

$$\sum_{k=0}^{\infty} C^k = (I - C)^{-1} \tag{84}$$

where I is the $N \times N$ identity matrix, the total number of sentences in the language $|L(G)|$ is just

$$|L(G)| = \vec{e}_1 (I - C)^{-1} \vec{f}^T \tag{85}$$

and the average sentence length $|\bar{W}|$ is given by

$$|\bar{W}| = \frac{\sum_k k N_k}{|L(G)|}. \tag{86}$$

If we now assume that $|L(G)| < \infty$ and the sentences are equiprobable, then we can compute the N_k, $|L(G)|$ and $|\bar{W}|$ from (83)-(85) which in turn allows us to compute the entropy $H_{eq}(L(G))$ from

$$H_{eq}(L(G)) = \frac{\log_2 |L(G)|}{|\bar{W}|}. \tag{87}$$

If the assumption of equiprobably sentences is not realistic, we can also compute the maximum entropy obtained under any probability distribution of sentences. In this case, we can show that [64]

$$H_{max}(L(G)) = -\log_2 (x_0) \tag{88}$$

where x_0 is chosen so that

$$1 - \sum_k N_k x_0^{\,k} = 0. \tag{89}$$

The acoustic word recognizer can be regarded as a noisy channel and, as such, can be characterized by its equivocation $H(x \mid y)$, which is a measure of the uncertainty in the spoken word x given the recognized word y. We can extend the notion of equivocation to sentences by defining the uncertainty in the spoken sentence given the acoustically recognized one according to

$$H(W \mid \tilde{W}) = \mid \overline{W} \mid H(x \mid y) \tag{90}$$

where

$$H(x \mid y) = \sum_{x \in V_T} \sum_{y \in V_T} p(x \mid y) p(x) \log_2 \left[\frac{p(x \mid y) p(x)}{p(y)} \right]. \tag{91}$$

The probabilities of confusion $p(x \mid y)$ can be experimentally measured; the prior probabilities of words $p(x)$ can be computed by the method given in [69]. Then the probability of occurrence of the word y at the output of the recognizer $p(y)$ can be calculated from

$$p(y) = \sum_{x \in V_T} p(x \mid y) p(x). \tag{92}$$

Finally, from an inequality due to Fano [70], it follows immediately that

$$H(W \mid \tilde{W}) \eta \leqslant \frac{H(P_e)}{\mid \overline{W} \mid} + P_e H_{eq}(L(G)) \tag{93}$$

where P_e is the probability of sentence error; $H(P_e)$ is defined as

$$H(P_e) = P_e \log_2 P_e + (1 - P_e) \log_2 (1 - P_e) \tag{94}$$

and the efficiency of the language η is given by

$$\eta = \frac{H_{eq}(L(G))}{\log_2 (\mid V_T \mid)}. \tag{95}$$

Since the entropy of the language is less than or equal to the logarithm of the vocabulary size, $\eta \leqslant 1$.

Equation (93) is not guaranteed to give a good estimate of P_e. However, for the speech-recognition system based on the flight information and reservation task language, the computation gives $p_e \geqslant 0.038$. The observed probability of error on 351 sentences was 0.039! The significance of (93) is that it enables us to estimate error probability in terms of measurable and computable properties of the task language and the acoustic recognizer. Since one has great flexibility in designing task languages for speech recognition, the grammars can be matched to word recognizers to yield robust communication.

C. Recognition of Connected and Continuous Speech

The discussion thus far has been predicated on the existence of a method of segmenting a sentence into its component words. In the experiments just described, the issue of segmentation was completely circumvented by requiring speakers to speak sentences with brief pauses between words. Sentences uttered in this manner can be segmented reliably on the basis of the temporal variation of energy in the speech waveform. Obviously, energy minima of sufficient duration signal word boundaries. With the location of the words so marked in the input utterance, isolated word recognizers of the type discussed in the first part of this paper serve well in the role of the acoustic processor shown in Fig. 33.

In continuous speech, however, no such pauses between words exist and segmentation becomes an error-prone process. In this section, we describe two basic methods for recognizing continuous speech in the presence of unreliable segmentation. The first is a syntax-directed method which matches isolated word templates to a continuous-speech stream. In this case, one copes with the segmentation problem by generating many hypothetical segmentations and choosing the best. The second method places the burden of correcting segmentation errors on the language analyzer by endowing it with the ability to insert missing segments and delete extra ones. A noteworthy aspect of this approach is that it has been implemented both for word-sized segments and more fundamental linguistic units, phonemes. The later method was used in a recognition system for fluent Japanese speech.

1) Syntax Directed Approach to Connected-Word Recognition: The origins of our syntax-directed architecture are found in some early attempts to improve the efficiency of the system for recognition of sentences composed of isolated words. Recall that in this system, the acoustic processor computes an entire D matrix. Since not all vocabulary words can occur in all word positions in the sentence, many of the entries of D are unused. The goal of these experiments then was to eliminate these unnecessary and costly distance computations. A solution to this problem was offered by Levinson and Rosenberg [71] in which a communication path was provided between the acoustic processor and the language analyzer (the dashed lines in Fig. 33). In this system, the language analyzer provides the acoustic processor with a list of those words which can occur in a given word position in the sentence being processed and the acoustic processor computes only those entries in the distance matrix. Besides providing a substantial reduction in computation, this "hypothesize and test" structure immediately suggests a method for connected-word recognition. We refer to this type of speech recognition as connected-word recognition because it is based on isolated-word templates and requires reasonable care of articulation although no pauses between words are required.

The system which was implemented by Levinson and Rosenberg [51] is shown in Fig. 36. The main components of the system are a parser similar to that described above, and an isolated-word recognizer which uses the UELM method of dynamic time registration discussed in Section II-B3).

Referring to the diagram of Fig. 36, the interaction between the parser and the time alignment subroutine is as follows. Based on its present state, the parser specifies that a vocabulary word, say v, should occur approximately at time t_b. The UELM algorithm matches the input speech beginning at t_b to a reference template for the word v. The distance of the match D_v is returned along with the computed endpoint t_e. This endpoint will then serve as a nominal beginning point for the next word in the candidate sentence.

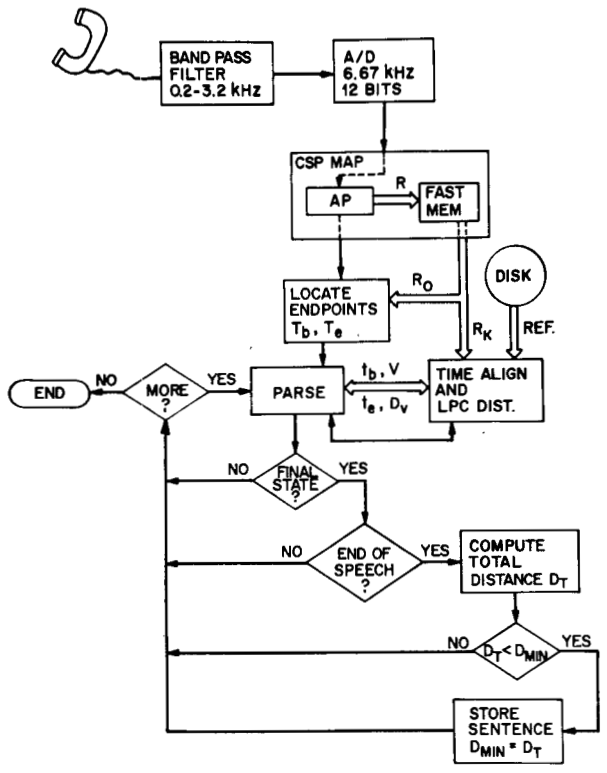

Fig. 36. Block diagram of the syntax-directed speech-recognition system.

The dynamic programming recursion by which the parser is implemented is identical to that of (78)-(80), except that a third matrix $[\Theta_{iq}]$ is added to keep track of the implicit segmentation performed by the UELM procedure. Initially, we set

$$\Theta_{iq} = \begin{cases} 1 & \text{for } i = 0, q = 1 \\ \infty & \text{otherwise} \end{cases} \tag{96}$$

where the initial values simply indicate that the parser must be in its initial state at the onset time of the speech input. Then we add the following to the basic recursion:

$$\Theta_{iq} = \Theta_{i-1,s^*} + \tau_{j^*} \tag{97}$$

where τ_{j^*} is the duration, in frames, of the segment of the input sentence to which the template for word v_{j^*} was aligned. Thus, when the parser continues a sentence from state q, it will begin the time alignment at frame $\Theta_{i+1,q}$.

After the evaluation of each word hypothesis, the parser checks to see if a final state has been reached. If the present state is not a final state, processing continues. Otherwise, a further check is performed to determine whether or not the last word terminated close to the endpoint of the input speech. If not, processing continues. Otherwise, a candidate sentence has been completed. If the newly generated sentence has a smaller metric than any previously produced, it is saved. Otherwise, it is discarded and processing continued.

The three step recursion of (79), (80), and (97) is performed by applying every transition in the state diagram once for each of $i = 0, 1, 2 \cdots$ thus building many candidate sentences in a strictly left to right fashion. Parsing continues until, for some i, every transition has been used without adding any word to any candidate sentence. Due to the rather liberal compression and expansion allowed by the time-alignment procedure, candidate sentences of a wide range of lengths are often generated. Usually, candidate sentences containing many more or may fewer words than were present in the input are poor ones. The final recognition of the input is the surviving candidate sentence when the termination criteria have all been satisfied.

The system has been tested on a set of 100 sentences ranging in length from 4 to 11 words and spoken by two male speakers. The median word-recognition rate was 90 percent and the sentence accuracy recorded was 75 percent. These results, which are significantly poorer than those obtained in the case of sentences with pauses between words, reflect the increased difficulty of the connected-speech task.

The principal shortcoming of the system can be seen from Fig. 37 in which are plotted the empirically determined probability density functions of the metrics for correct words (HITS), $p_H(d)$, and incorrect ones (ERRORS), $p_E(d)$. The equal error threshold occurs at $d = 0.528$ and the corresponding error probability is 0.275. Thus it may be seen that while the probability is high that a word having a small distance is correct, there is a substantial likelihood that a correct word will have a large distance. Since the parser maintains only the best path to each state, one such large distance can cause the parser to abandon an otherwise correct path. Since the segmentation will then not match that of the true sentence, the correct path will not usually be recovered. The obvious solution is to retain more than one candidate sentence for each state.

2) Explicit Segmentation and Labeling for Continuous-Speech Recognition: The principal advantage of the syntax-directed approach to connected-word recognition is that by using word size templates we implicitly have good representations of coarticulation effects within word boundaries. Unfortunately, the coarticulation effects across word boundaries in fluent speech are not well accounted for by reference patterns derived from isolated words. One approach to this problem is to use an explicit segmentation and labeling scheme in conjunction with a parser which can correct the segmentation errors which will certainly occur.

To illustrate the power of such a parser, consider the following example derived from the toy language shown in Fig. 34. Suppose that the sentence "I NEED A FIRST CLASS SEAT" was incorrectly segmented with the article "A" being deleted. But suppose that all other words were recognized correctly with low distances. A parser which assumes correct segmentation would interpret the input as either "I NEED A NONSTOP FLIGHT" or some other five word sentence depending on the values in the distance matrix. In any event, it cannot get the correct sentence because it has six segments (words) and the parser cannot alter the number of segments. The parser we shall describe in this section has the desirable property of being able to insert and delete segments, and, by that means, will derive the correct sentence from the erroneously segmented transcription of our example.

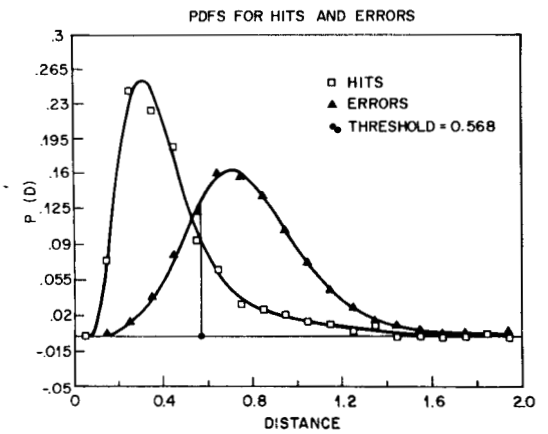

Fig. 37. Probability density functions for correct and incorrect words in connected-speech recognition.

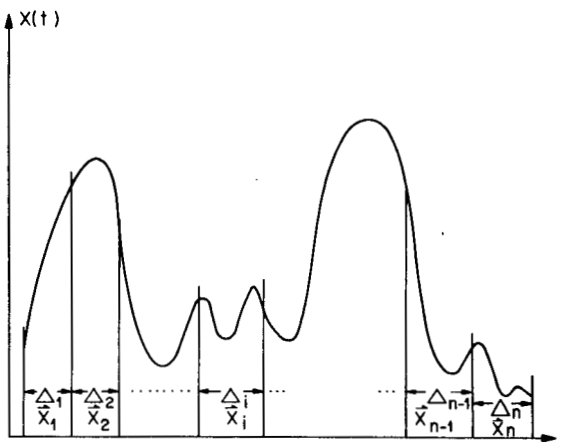

Fig. 38. Example illustrating segmentation of an acoustic waveform.

The ability of a parser to function in the presence of segmentation errors is especially important for large vocabulary continuous-speech recognition tasks because, as the vocabulary grows, it becomes unfeasible to have reference templates for each word. Rather, one would like to use a smaller set of "universal" templates in terms of which one can construct the entire lexicon of a chosen natural language. One such set is the set of phonemes which are the "atomic sounds" of spoken language. It is here that algorithmic parsers, such as those we are discussing, shine, because their operation is independent of the grammar and/or symbols used. In this section, we shall describe just such a parser for fluent Japanese based on a 27-phoneme representation of that language.

The parser which we describe below requires the acoustic processor to segment and label the speech waveform as shown in Fig. 38. The acoustic processor must place segment boundaries in the speech stream so that the segments, $\Delta_1, \Delta_2, \cdots, \Delta_k$, correspond to the vocabulary symbols which the recognizer and parser know. These segments, which are not necessarily of equal length, are then analyzed and feature vectors $\vec{x}, \vec{x}_2, \cdots, \vec{x}_k$ are extracted. The feature vectors are then used to compute the entries of the D matrix in which there will be one row for each segment of the waveform. The difference between this distance matrix and the one used earlier is that previously each row of D corresponded to a real-word position in the sentence; whereas, in this case, the segments are not certain to correspond one-for-one to symbols. Some segments may be superfluous while others may actually contain several symbols. In this segmentation and labeling paradigm, the parser must be able to correct such segmentation errors.

The parser, which is described in detail by Levinson [72] requires two dynamic programming algorithms, one nested inside the other. The outer and main one is similar to a procedure due to Wagner [73] which was used to correct spelling errors in text. The second and inner procedure uses the algorithm of Dijkstra cited earlier.

The outer dynamic program uses two rectangular matrices $[\Phi_{iq}]$ and $[\Psi_{iq}]$ where $1 \leqslant i \leqslant k$, $1 \leqslant q \leqslant N$ and k is the number of rows of D. These two matrices are initialized exactly as in (78a) and (78b). The basic recursion is then given by

$$\Phi_{iq} = \min_{1 \leqslant s \leqslant N} \{\Phi_{i-1,s} + E_{sqi}\} \tag{98a}$$

and

$$\Psi_{iq} = \Psi_{1-1,s_0} * \beta^* \tag{98b}$$

where E_{sqi} is the length, in terms of D, of the shortest path from state s to state q based on the ith segment, β^* is the corresponding sentence fragment, and $s_0{}^*$ is the state for which (98a) is minimized.

Equations (98a) and (98b) have a very simple interpretation. They say that the best path from state 1 to state q based on the ith segment of the input is just the best path from state 1 to state s concatenated to the best path from state s to state q. It is important to note that there are no restrictions on the number of symbols in β; therefore the ith segment may account for zero symbols, in which case the segment is deleted; one symbol meaning that the segment is correct; or many symbols in which case all symbols except one are inserted.

To actually calculate E_{qsi} and β, we use Dijkstra's algorithm; there is, however, an added complication. If the length of β is greater than one then one symbol comes from the ith segment and all others are inserted. Since we do not know which symbol is the real one, all possibilities must be tried. For this reason, the algorithm requires three dimensional arrays $[\phi_{rlp}]$ and $[\psi_{rlp}]$. Initially

$$\phi_{rlp} = \begin{cases} 0 & \text{for } l = 0, p = s \text{ and } \forall r \\ \infty & \text{otherwise} \end{cases} \tag{99a}$$

and

$$\psi_{rlp} = 0 \qquad \forall r, l, p. \tag{99b}$$

Then the basic recursion proceeds according to

$$\phi_{rmy} = \min_{\Delta} \{\phi_{r,m-1,t} + h_{ij_m}\} \tag{100}$$

where Δ is the set of all transitions from state t to state y with

a label of v_j and

$$h_{ij_m} = \begin{cases} d_{ij_m} & \text{for } m = r \\ \xi(d_{ij_m}) & \text{for } m \neq r. \end{cases} \tag{101}$$

That is, when $m = r$ we are taking the symbol from the ith segment and thus using its distance from the D matrix. In all other cases, we are inserting a symbol and its score is some function of that entry $\xi(d_{ij_m})$. The other part of the recursion is given by

$$\psi_{rmy} = \psi_{r,m-1,t^*} \cdot v_j^* \tag{102}$$

where t^* and v_j^* minimizes (100). Then

$$E_{sqi} = \min_{l} \left\{ \min_{1 \leq r \leq l} \{\phi_{rlq}\} \right\} \tag{103}$$

and

$$\beta = \psi_{r^* l^* q} \tag{104}$$

where r^* and l^* minimize (103).

Finally, substituting E_{sqi} from (102) into (98a) and β from (104) into (98b) we get

$$D(\hat{W}) = \min_{z} \{\Phi_{kz}\} \tag{105}$$

where the minimization is over all final states z. In some cases it is desirable to bias $\hat{D}(W)$ in favor of longer sentences by normalizing by the length of Ψ_{kz}. The optimal sentence $\hat{W}$ is determined from

$$\hat{W} = \Psi_{kz^*}$$

where z^* minimizes (105).

The speech-recognition task on which the parser was tested is described in detail by Shikano and Kohda [53] and Nakatsu and Kohda [54]. We give only a brief description here. The task of the system is railroad seat reservations. An input consists of a sequence of up to eight phrases spoken in fluent Japanese and separated by pauses of approximately 0.5 s duration. Each phrase except one specifies one item of information necessary to obtain a reservation. The other phrase is the verb of the sentence comprising all of the phrases. The phrases need not occur in any order, except that the verb, if present, must be last; nor must all phrases be present. The sentences are composed from a 112-word vocabulary while the words are represented in terms of just 27 phonemes.

The acoustic processor segments each phrase first into syllables and then phonemes. The distance matrix is derived from this second segmentation and passed to the parser. The parser analyzes each phrase in terms of each of eight grammars. The best phrases from each grammar are selected and their scores entered in another distance matrix $\boldsymbol{D}$ defined by

$$d_{ij} = D(\hat{W}_j \mid G_i) 1 \leq i \leq 8, 1 \leq j \leq N_w \tag{107}$$

where $\hat{D}(W_j \mid G_i)$ denotes the minimum distance for the jth phrase in the sentence when parsed into the ith grammar. The number of phrases in the sentence is represented by N_w.

Since at most one phrase can be generated by any grammar and the verb, if present must be last, we minimize the total phrase distance

$$T = \sum_{i=1}^{N_w} d_{ijj} \tag{108}$$

subject to the constraints that $i_k \neq i_l$ for $k \neq l$ and $i_j = 3$ only for $j = N_w$. The latter constraint is because G_3 is the grammar for verb phrases. The minimization is accomplished by means of the Cartesian product algorithm described earlier and from it, the maximum-likelihood sentence is derived.

The system was tested on 20 sentences comprising 124 phrases spoken by each of four male subjects. The accuracy of the acoustic recognition which formed the input to the parser was as follows. Phonemes were correctly classified 64 percent of the time. Of the errors, 23 percent were due to simple substitutions. The remaining 16 percent involved segmentation faults. Under these conditions, an overall correct phrase-recognition rate of 68 percent was achieved. In absolute terms, this constitutes an unacceptably high error rate, but in comparison with the error rate which is observed, using the acoustic processor alone it becomes a striking demonstration of the power of linguistic analysis in speech recognition.

D. Modeling the Speech-Communication Process

Although the ultimate purpose of speech-recognition research is to enable natural spoken language human/machine communication, the systems described thus far simply transcribe speech. In this section we describe a complete system for human/machine speech communication. The system utilizes acoustic, syntactic, and semantic processing to effect a true dialog. The user of the system must speak in sentences with pauses between words and the machine responds in a similar fashion. The system is highly robust and operates on line in a few times real time on a laboratory minicomputer. Details of the system as implemented for the airline timetable task, are given by Levinson and Shipley [74]. The following overview, however, will be useful for these discussions.

A block diagram of the system is shown in Fig. 39. From the figure we see that speech in the form of a sentence W is input to the word recognizer which operates in conjunction with the parser in the syntax directed mode discussed earlier. The output of the parser is the maximum-likelihood sentence $\hat{W}$ and its derivation or state sequence $\bar{q}$. The semantic processor takes $\bar{q}$ and $\hat{W}$ and interprets them in the context of the conversation stored in the u-model (universe) to generate "actions" which involve searching the Official Airline Guide data base, altering the context of the conversation, and formulating an appropriate response.

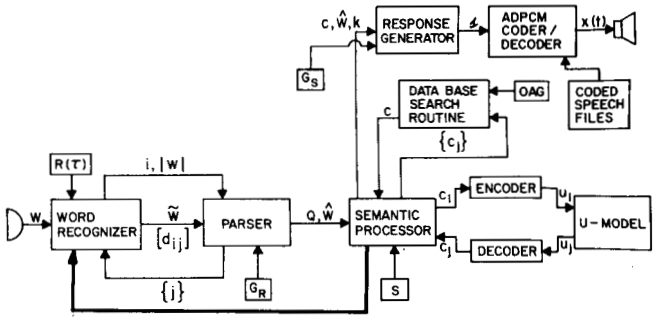

Fig. 39. Block diagram of the conversational-mode speech-recognition system.

The data base search routine can take a set of items $\{c_j\}$ and either match it to a complete flight description C or determine in what way the set is insufficient. A complete flight description can be used to answer a question and/or update the u-model.

The response generator takes the current flight description C, the input sequence $\hat{W}$, and a semantic code K, and generates a reply to $\hat{W}$ using the grammar G_s. The reply is first represented by a string of symbols each of which is a code for one word of a sentence S. A subroutine controls an ADPCM coder/decoder, uses S to access a file of precoded isolated words, and concatenates them into a spoken rendition of the reply.

1) Semantic Processing: The semantic analysis which is performed by the system is based on the following abstraction of the communication process. For A to communicate with B, both must have a model or internal representation of the subject. A takes the state of his model and encodes it in a message which he transmits to B. B decodes the message in terms of his subject model and alters its state accordingly. Communication takes place to the extent that B's model is isomorphic to the state A's would be in had he received the same message.

This abstraction poses two requirements. First we must have some way of defining states of a "world-model" and second, there must be a mechanism for changing states. In our system, the states are implicitly defined by the semantic structure of the task language. The language has 19 semantic categories such as *information, reservation, flight choice, seat selection,* etc. These define 15 parameters of the task model such as *destination, day of the week, departure time,* etc. The particular values which are assigned to the parameters determine the state of the model, for example, *destination* = BOSTON, *day of the week* = TUESDAY, *departure time* = MORNING. The state of the U-model is changed by instructions called "actions" which mediate between the input, the data base, and the task model.

Semantic processing, then, occurs in two stages. First, there is a decoding stage in which it is determined from the input which "actions" are to be executed. Then the "actions" must actually be performed. Semantic decoding is simply a mapping

$$S: \{Q \times V_T\} \rightarrow A \tag{109}$$

where Q is the set of states of the diagram of G and A is the set of "actions" which we shall define precisely below. Thus, if $\hat{W} = \hat{w}_1 \hat{w}_2 \cdots \hat{w}_k$ is a sentence which has been recognized by the acoustic processor and parser and $\bar{q} = q_0 q_1 \cdots q_k$ is its state sequency then

$$S(q_i, \hat{w}_i) = \alpha_i \in A \text{ for } 1 \leqslant i \leqslant k. \tag{110}$$

The "actions" α_i are nothing more than instructions for the classical finite state machine. Thus an action is just the four-tuple

$$\alpha_i = \alpha_i(X, U_p, K, U_N) \tag{111}$$

which is understood to mean, if the task model is presently in state U_p and the input contains the set of words $X \in V_T^*$, then change the state of the task model to U_N and issue output code K which will be used by the response generator to construct the reply $R(\hat{W})$.

The semantic processor has five mechanisms for changing states in the U-model: default values, data base lookup, direct translation from the input, derivation from the present state, and computation based on information and procedures not reflected in the current state of the U-model.

There are 126 entries in the semantic decoding table S, and 37 instructions in the "action" set A. These are stored in a static portion of memory while the U-model and the subroutines, which are special purpose procedures for changing its state, are stored in dynamic memories.

The following sample dialog gives an idea of the level of sophistication of the conversations in which the system can participate.

Machine:	This is the Bell Laboratories flight information system. May I help you?
User:	I want one first class seat on flight number three one to Denver on Sunday.
M:	I am sorry, the flight is not available on Sunday.
U:	I would like to leave on Saturday.
M:	Flight number three one leaves New York at nine forty five A.M.; arrives in Denver at one ten P.M.
U:	What is the flight time?
M:	The flight time is five hours and twenty five minutes.
U:	How many stops are there on the flight?
M:	This flight makes one stop.
U:	I want a nonstop flight.
M:	I am sorry there are no nine A.M. nonstop flights.
U:	I want to go at four P.M.
M:	Flight number one six leaves New York at four fifty P.M.; arrives in Denver at seven P.M.
U:	What stops are there on the flight?
M:	This flight makes no stops.
U:	Is a meal served on the flight?
M:	Dinner is served.
U:	How much is the fare?
M:	The round-trip first-class fare is three hundred sixty dollars.
U:	What plane is on the flight?
M:	The aircraft is a Douglas DC ten.

U: I prefer the Boeing seven four seven.
M: The aircraft is a Douglas DC ten.
U: Please repeat the departure time.
M: The departure time is four fifty P.M.
U: I will take flight number six five.
M: I am sorry there is no flight six five to Denver.
U: Please repeat the flight number.
M: The flight number is one six.
U: I want flight one six.
M: Your reservation on flight number one six to Denver on Saturday is confirmed. Thank you.

The dialog is stilted but communication is effected. The important result derived from this experimental system is that by combining several simple systems, a synergistic effect is achieved in which the combination can perform tasks of a sophistication greater than would be expected on the basis of the complexity of the components.

The system is also highly robust. It was evaluated on a corpus of 351 sentences spoken by five male and two female speakers. After syntactic processing, 21 sentences contained word-recognition errors. However after semantic analysis 15 of these were correctly understood despite the errors and correct responses were made. In the remaining six cases, a semantic anomaly was detected causing the machine to pose a question to the user who could then correct the problem. Thus we achieved a recognition accuracy of more than 98 percent and when recognition errors occurred, communication did not break down. Thus, we see that in making even this crude model of speech communication, a surprising level of robustness is obtained. This indicates that to make progress in speech recognition one needs not only to improve the individual components, but also to treat the problem in total and design better control structures to coordinate the components.

V. CONCLUSION

In the previous section of the paper, we have discussed the application of higher level processing to the problem of speech recognition. We discussed grammatical analysis from the standpoint first of list searching and then parsing within a formal language theoretical framework. This formulation lead to an analytical understanding of how syntactic information makes speech communication more robust. We have described four experimental systems which utilize syntactic information to improve speech-recognition accuracy. We then discussed semantic processing and showed how it could be used to build a conversational-mode speech-recognition system. A summary of these five systems and some of their pertinent specifications is shown in Table VI. The main conclusion that can be drawn from this table is that as better high-level processing is included, thereby simulating cognitive and intelligence related processes, more sophisticated and robust human/machine communication is obtained.

At the end of such a presentation, it is appropriate to address the question of directions for further research. This is a highly speculative venture and workers in the field have had little success in predicting what would be an instructive next step. In 1973, Newell *et al.* [75] argued cogently that the next frontier involved the intelligence related and structural aspects of speech. Their thesis was that only higher level processing could resolve ambiguities in the necessarily error-ridden acoustic transcription. Thus was born the ARPA project. In 1976, after termination of the herculean effort, Klatt [24] assessed the project and concluded that the most successful system was the HARPY machine of Lowerre [55], a system of conventional design using the classical syntax analysis described in this paper. Although many insights were gained through the HEARSAY [20] and HWIM [21] systems, they did not meet the original design specification. In a recent survey of speech-recognition research, Lea and Shoup [76] concluded that the new frontier is acoustic and prosodic analysis. So we have made a complete cycle.

We suspect that what is truly needed is a better understanding of all levels of the speech communication process from acoustics and phonetics to pragmatics. We suggest that some potentially fruitful investigations into certain aspects of the speech communication process have been conspicuously absent from the research forum. These are the study of adaptive schemes, the relation of speech recognition to speech synthesis and the incorporation of models of hearing and perception in acoustic analysis.

In humans, speech is an acquired ability not present at birth. Perhaps it is too difficult a process to emulate by a deterministic program but rather must be learned by adapting to an environment. The detailed Markov model and careful statistical analysis which is unique to the speech-recognition system of Jelinek [56] and Bahl *et al.* [77] is probably well suited to be part of an adaptive or self-organizing strategy.

Presently there are several systems which synthesize speech from text [59], [78], [79]. The state of the art in synthesis is advanced in comparison to that of recognition. Flanagan [80] has suggested that the discrepancy in performance indicates that there is a great deal of information about speech embedded in these synthesis systems that is presently not, but should be, used in recongition.

Finally, from models of the peripheral auditory system of Allen [81] and Hall [82] it is becoming clear that the ear performs a kind of spectral analysis which cannot be approximated by conventional methods. Perhaps these models can provide the extreme selectivity and nonlinear effects which are needed to improve acoustic analysis.

As for the ultimate limits of our ability to provide spoken natural language human/machine communication, we tend to be conservative in our estimates. We believe that disciplined discourse in artificial languages about carefully circumscribed topics with computers will be possible and economically viable in the not-too-distant future. At the present moment, the worthy goal of unconstrained speech communication seems quite beyond our reach.

ACKNOWLEDGMENT

The authors would like to acknowledge the contributions of several of their colleagues to the work described in this paper. Dr. James L. Flanagan has been the leader of the effort to bring speech-recognition research to prominence at Bell Laboratories,

TABLE VI
SUMMARY OF PERFORMANCE OF SEVERAL SPEECH-RECOGNITION SYSTEMS

Task	Input Speech	Vocabulary	Type of Processing	Accuracy	Remarks
Airline Timetable Information and Reservations	Sentences With Pauses Between Words	127 Words	Isolated Word Recognition With Maximum Likelihood Parsing	> 96% on Sentences	Reference [66] Speaker Trained
Telephone Directory Assistance	Spelled Names With Pauses Between Letters	26 Letters	Maximum Likelihood List Searching Isolated Word Recognition	> 98% Correct Information Retrieval	Uses 18,000 Name BTL Directory Speaker Independent References [46,49]
Airline Timetable Information and Reservations	Carefully Articulated Sentences, No Pauses Between Words	127 Words	Syntax-Directed Dynamic Programming Time Registration	75% Sentences 90% Words	Speaker Trained Reference [71]
Train Seat Reservations	Fluent Japanese Phrases	112 Words 27 Phonemes	Segmentation and Labelling Maximum Likelihood Parsing Corrects Segmentation Errors	68% Phrases	Speaker Independent References [72,53,54]
Airline Timetable Information and Reservations	Sentences With Pauses Between Words	127 Words	Isolated Word Recognition Maximum Likelihood Parsing Semantic Analysis	> 98% Sentences and Error Detection	Speaker Trained Reference [74] Conducts Spoken Dialog Official Airline Guide Data Base

and he has provided guidance, insight, and numerous valuable suggestions and advice. Dr. Aaron Rosenberg has been involved with almost every aspect of the recognition problem and has made substantial research contributions. Dr. Fumitada Itakura was responsible for the earliest recognition systems upon which this paper is based. Cory Myers has been primarily responsible for much of our understanding of dynamic time-warping algorithms and their application to isolated and connected word recognition. We also thank Cory for his diligent reading and comments on the paper. Finally, we acknowledge the contributions of Jay Wilpon in the implementation of many of the systems described in this paper.

REFERENCES

[1] J. L. Flanagan, *Speech Analysis, Synthesis, and Recognition*, 2nd ed. New York: Springer-Verlag, 1972.
[2] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
[3] N. R. Dixon and T. B. Martin, Eds., *Automatic Speech and Speaker Recognition*. New York: IEEE Press, 1979.
[4] W. Lea, Ed., *Trends in Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1980.
[5] D. R. Reddy, Ed., *Speech Recognition*. New York: Academic, 1974.
[6] T. B. Martin, "Practical applications of voice input to machines," *Proc. IEEE*, vol. 64, pp. 487–501, Apr. 1976.
[7] S. Moshier, "Talker independent speech recognition in commercial environments," in *Speech Commun. Papers, 97th ASA Meeting*, June 1979, pp. 551–553.
[8] H. Sakoe, "Two-level DP-matching—A dynamic programming based pattern matching algorithm for connected word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 588–595, Dec. 1979.
[9] Interstate Electronics Corp., Voice Data Entry System, unpublished tech. descriptions.
[10] Centigram Corp., Mike, unpublished tech. descriptions.
[11] Heuristics Corp., Speechlab, unpublished tech. description.
[12] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 67–72, Feb. 1975.
[13] A. E. Rosenberg and F. Itakura, "Evaluation of an automatic word recognition system over dialed-up telephone lines," *J. Acoust. Soc. Amer.*, suppl. 1, vol. 60, p. S12, 1976.
[14] M. R. Sambur and L. R. Rabiner, "A speaker-independent digit-recognition system," *Bell Syst. Tech. J.*, vol. 54, pp. 81–102, Jan. 1975.
[15] L. R. Rabiner, "On creating reference templates for speaker independent recognition of isolated words, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 34–42, Feb. 1978.
[16] L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker-independent recognition of isolated words using clustering techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 336–349, Aug. 1979.
[17] J. S. Bridle and M. D. Brown, "Connected word recognition using whole word templates," in *Proc. Inst. Acoust.*, Autumn 1979.
[18] L. R. Rabiner and C. E. Schmidt, "Application of dynamic time warping to connected digit recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 337–388, Aug. 1980.
[19] C. S. Myers and L. R. Rabiner, "Connected digit recognition using a level building DTW algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, June 1981.
[20] V. R. Lesser, R. D. Fennell, L. D. Erman, and D. R. Reddy, "Organization of the hearsay II speech understanding system," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 11–24, Feb. 1975.
[21] J. J. Wolf and W. A. Woods, "The HWIM speech understanding system," in *Conf. Rec. 1977 IEEE Int. Conf. Acoust., Speech, Signal Processing*, May 1977, pp. 784–787.
[22] S. E. Levinson, A. E. Rosenberg, and J. L. Flanagan, "Evaluation of a word recognition system using syntax analysis," *Bell Syst. Tech. J.*, vol. 57, pp. 1619–1626, May–June 1978.
[23] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a linguistic decoder for the recognition of continuous speech," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 250–256, May 1975.
[24] D. H. Klatt, "Review of the ARPA speech understanding project," *J. Acoust. Soc. Amer.*, vol. 62, pp. 1345–1366, Dec. 1977.
[25] J. S. Bridle and M. D. Brown, "An experimental automatic word-recognition system: Interim report," JSRU Res. Rep. 1003, Dec. 1974.
[26] G. M. White and R. B. Neely, "Speech recognition experiments

with linear prediction, bandpass filtering, and dynamic programming," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 183–188, Apr. 1976.

[27] H. F. Silverman and N. R. Dixon, "A comparison of several speech-spectra classification methods," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 289–295, Aug. 1976.

[28] D. R. Reddy, L. D. Erman, and R. B. Neely, "A model and a system for machine recognition of speech," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 229–238, June 1973.

[29] J. D. Markel and A. H. Gray Jr., *Linear Prediction of Speech*. New York: Springer-Verlag, 1976.

[30] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561–580, Apr. 1975.

[31] G. M. White, "Dynamic programming, the Viterbi algorithm, and low cost speech recognition," in *Proc. 1978 IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 413–417, Apr. 1978.

[32] A. H. Gray, Jr. and J. D. Markel, "Distance measures for speech processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 380–391, Oct. 1976.

[33] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 43–49, Feb. 1978.

[34] C. S. Myers, L. R. Rabiner, and A. E. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 622–635, Dec. 1980.

[35] L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson, "Considerations in dynamic time warping for discrete word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 575–582, Dec. 1978.

[36] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[37] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[38] L. F. Lamel, "Methods of endpoint detection for isolated word recognition," M.S. thesis, Massachusetts Inst. Technol., Feb. 1980.

[39] L. R. Rabiner and J. G. Wilpon, "Application of clustering techniques to speaker-trained isolated word recognition," *Bell Syst. Tech. J.*, vol. 58, pp. 2217–2233, Dec. 1979.

[40] ——, "Speaker independent isolated word recognition for a moderate size (54 word) vocabulary," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 583–587, Dec. 1979.

[41] L. R. Rabiner, J. G. Wilpon, and A. E. Rosenberg, "A voice-controlled repertory-dialer system," *Bell Syst. Tech. J.*, vol. 59, pp. 1153–1163, Sept. 1980.

[42] L. R. Rabiner and J. G. Wilpon, "Application of isolated word recognition to large vocabularies," unpublished.

[43] ——, "A two pass system for isolated word recognition," *Bell Syst. Tech. J.*, vol. 60, pp. 739–766, May–June 1981.

[44] J. G. Ackenhusen and L. R. Rabiner, "Microprocessor implementation of an LPC-based isolated word recognizer," *BTL Microprocessor Symp.*, Sept. 1980.

[45] A. E. Rosenberg and C. E. Schmidt, "Automatic recognition of spoken spelled names for obtaining directory listings," *Bell Syst. Tech. J.*, vol. 58, pp. 1797–1823, Oct. 1979.

[46] B. Aldefeld, L. R. Rabiner, A. E. Rosenberg, and J. G. Wilpon, "Automated directory listing retrieval system based on isolated word recognition," *Proc. IEEE*, vol. 68, pp. 1364–1379, Nov. 1980.

[47] P. Cummiskey, N. S. Jayant, and J. L. Flanagan, "Adaptive quantization in differential PCM coding of speech," *Bell Syst. Tech. J.*, vol. 52, pp. 1105–1118, Sept. 1973.

[48] L. R. Rabiner and R. W. Schafer, "Digital techniques for computer voice response: Implementations and applications," *Proc. IEEE*, vol. 64, pp. 416–433, Apr. 1976.

[49] B. Aldefeld, S. E. Levinson, and T. G. Szymanski, "A minimum distance search technique and its application to automatic directory assistance," *Bell Syst. Tech. J.*, vol. 59, pp. 1343–1356, Oct. 1980.

[50] C. S. Myers and L. R. Rabiner, "A dynamic time warping algorithm for connected word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, Apr. 1981.

[51] S. E. Levinson and A. E. Rosenberg, "A new system for continuous speech recognition—Preliminary results," in *Proc. ICASSP-79*, pp. 239–243, Apr. 1979.

[52] P. B. Denes, "The design and operation of a mechanical speech recognizer at University College, London," *J. Brit. IRE*, vol. 19, pp. 219–229, Apr. 1959.

[53] K. Shikano and M. Kohda, "A linguistic processor in a conventional speech recognition system," *Rev. ECL*, vol. 26, pp. 1486–1504, 1978.

[54] R. Nakatsu and M. Kohda, "An acoustic processor in a conversational speech recognition system," *Rev. ECL*, vol. 26, pp. 1505–1520, 1978.

[55] B. T. Lowerre, "The HARPY speech recognition system," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1976.

[56] F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, vol. 64, pp. 532–556, Apr. 1976.

[57] C. S. Peirce, *Collected Papers of Charles Sanders Peirce*, C. Hartstone and P. Weirs, Eds. Cambridge, MA: Harvard Univ. Press, 1935.

[58] C. H. Coker, "A model of articulatory dynamics and control," *Proc. IEEE*, vol. 64, pp. 452–460, Apr. 1976.

[59] C. H. Coker, N. Umeda, and C. P. Browman, "Automatic synthesis from ordinary english text," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 293–298, Feb. 1973.

[60] J. L. Flanagan, K. Ishizaka, and K. Shipley, "Synthetic speech from a dynamic model of the vocal cords and vocal tract," *Bell Syst. Tech. J.*, vol. 54, pp. 485–506, Mar. 1975.

[61] A. V. Aho, T. G. Szymanski, and M. Yannakakis, "Enumerating the Cartesian product of ordered sets," in *Proc. 14th Annu. Conf. Information Sciences and Systems*, Princeton, NJ, Mar. 1980.

[62] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*. Reading, MA: Addison-Wesley, 1973, pp. 422 ff.

[63] N. Chomsky, "On certain formal properties of grammars," *Inform. Contr.*, vol. 2, pp. 137–167, 1959.

[64] J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*. Reading, MA: Addison-Wesley, 1969.

[65] E. W. Dijkstra, "A note on two problems in connection with graphs," *Num. Math.*, vol. 1, pp. 269–271, 1959.

[66] S. E. Levinson, "The effects of syntactic analysis on word recognition accuracy," *Bell Syst. Tech. J.*, vol. 57, pp. 1627–1644, May–June 1977.

[67] S. E. Levinson, R. J. Lipton, and L. Snyder, "Some results on maximum *a posterori* probability parsing," in *Proc. 1976 Conf. Inform. Sciences and Systems*, Baltimore, MD, Apr. 1976.

[68] C. E. Shannon, "Prediction and entropy of printed english," *Bell Syst. Tech. J.*, vol. 1, pp. 50–64, Jan. 1951.

[69] M. M. Sondhi and S. E. Levinson, "Computing relative redundancy to measure grammatical constraint in speech recognition tasks," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Tulsa, OK, Apr. 1978, pp. 409–412.

[70] R. M. Fano, *Transmission of Information, A Statistical Theory of Communications*. New York, NY: M.I.T. Press and Wiley, 1961, pp. 186 ff.

[71] S. E. Levinson and A. E. Rosenberg, "Some experiments with a syntax directed speech recognition system," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Tulsa, OK, Apr. 1978, pp. 700–703.

[72] S. E. Levinson, "An algorithm for maximum likelihood parsing of speech in the presence of segmentation errors and an experimental performance evaluation," *IEEE Trans. Acoust., Speech, Signal Processing*, to be published.

[73] R. A. Wagner, "Order-*n* correction for regular languages," *Commun. Ass. Comput. Mach.*, vol. 17, pp. 265–269, May 1974.

[74] S. E. Levinson and K. L. Shipley, "A conversational mode airline information and reservation system using speech input and output," *Bell Syst. Tech. J.*, vol. 59, pp. 119–137, Jan. 1980.

[75] A. Newell, J. Barnett, J. W. Forgie, C. C. Green, D. H. Klatt, J. C. R. Licklider, J. Munson, D. R. Reddy, and W. A. Woods, *Speech Understanding Systems: Final Report of a Study Group*. Amsterdam, The Netherlands: North Holland/American Elsevier, 1973.

[76] W. A. Lea and J. E. Shoup, "Gaps in the technology of speech understanding," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Tulsa, OK, Apr. 1978, pp. 405–408.

[77] L. R. Bahl, R. Bakis, P. S. Cohen, A. G. Cole, F. Jelinek, B. L. Lewis, and R. L. Mercer, "Further results on the recognition of a

continuously read natural corpus," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Denver, CO, Apr. 1980, pp. 872–875.
[78] O. Fujimura, M. J. Macchi, and J. B. Lovins, "Demisyllables and affixes for speech synthesis," in *Proc. 9th Int. Congress Acoust.*, vol. 1 (Madrid, Spain), July 1977, p. 513.
[79] J. Allen, "Synthesis of speech from unrestricted text," *Proc. IEEE*, vol. 64, pp. 433–442, Apr. 1976.
[80] J. L. Flanagan, "Computers that talk and listen: Man–machine communication by voice," *Proc. IEEE*, vol. 64, pp. 405–415, Apr. 1976.
[81] J. B. Allen, "Cochlear micromechanics—A mechanism for transforming mechanical to neural tuning within the cochlea," *J. Acoust. Soc. Amer.*, vol. 62, pp. 930–939, 1977.
[82] J. L. Hall, "Two tone suppression in a non-linear model of the basilar membrane," *J. Acoust. Soc. Amer.*, vol. 61, pp. 802–810, 1977.
[83] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions and substitutions with applications to speech recognition," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 404–411, 1975.
[84] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.
[85] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM J. Res. Development*, vol. 13, pp. 675–678, 1969.

Lawrence R. Rabiner (S'62–M'67–SM'75–F'75) was born in Brooklyn, NY, on September 28, 1943. He received the S.B. and S.M. degrees simultaneously in June 1964, and the Ph.D. degree in electrical engineering in June 1967, all from the Massachusetts Institute of Technology, Cambridge, MA.

From 1962 through 1964, he participated in the cooperative plan in electrical engineering at Bell Laboratories, Whippany, NJ, and Murray Hill, NJ. He worked on digital circuitry, military communications problems, and problems in binaural hearing. Presently, he is engaged in research on speech communications and digital signal-processing techniques at Bell Laboratories, Murray Hill. He is coauthor of the books *Theory and Application of Digital Signal Processing* (Prentice-Hall, 1975) and *Digital Processing of Speech Signals* (Prentice-Hall, 1978).

Dr. Rabiner is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, and a Fellow of the Acoustical Society of America. He is a former President of the IEEE S-ASSP Ad Com, and is currently a member of the S-ASSP Technical Committee on Digital Signal Processing, former member of the S-ASSP Technical Committee on Speech Communication, former Associate Editor of the S-ASSP TRANSACTIONS, member of the PROCEEDINGS OF THE IEEE Editorial Board, and a former member of the Technical Committee on Speech Communication of the Acoustical Society.

★

Stephen E. Levinson (S'72–M'74) was born in New York, NY, on September 27, 1944. He received the B.A. degree in engineering sciences from Harvard University, Cambridge, MA, in 1966, and the M.S. and Ph.D. degrees in electrical engineering from the University of Rhode Island, Kingston, RI, in 1972 and 1974, respectively.

From 1966–1969, he was a Design Engineer at Electric Boat Division of General Dynamics in Groton, CT. From 1974–1976, he held a J. Willard Gibbs Instructorship in Computer Science at Yale University. In 1976, he joined the technical staff at Bell Laboratories, Murray Hill, NJ, where he is pursuing research in the areas of speech recognition and cybernetics.

Dr. Levinson is a member of the Association for Computing Machinery and the Acoustical Society of America, and is Chairman of the IEEE Computer Society Technical Committee on Speech Recognition and Understanding.